Kaunas University of Technology

Faculty of Informatics

T120B162 Software Systems Testing

Todžės Warriors

Lab 3. Static Testing

Ugnius Rinkevičius	**Student**
Martynas Kuliešius	**Student**
Nedas Liaudanskis	**Student**
Paulius Osipauskas	**Student**

Prof. Lukas Arlauskas

Table of content

Lab. 3. Static code analysis

Introduction

The purpose of this laboratory work is to perform a code review and apply a static analysis tool to the developed SignalR Snake 2D game. During the analysis process, the aim will be to identify potential errors and correct them. Additionally, create a new rule for the static analysis tool and apply it to the project's code.

Code Review

1. Classes selected for review:
    - Snakehub.cs - Main SignalR hub for managing snake game actions.
    - Snake.cs - Represents individual snake objects in the game.
    - Field.cs - Represents the game field
2. Possible issues:
    a. **Structure Violations:** Inconsistent variable naming, potential Single Responsibility Principle (SRP) violations, and lack of encapsulation in `Field`.
    b. **Performance bottlenecks**: loops in `MoveTimer_Elapsed` can be optimized to avoid redundant operations.
    c. **Defensive Programming:** Lack of checks for null or invalid states in critical methods.
    d. **Documentation Gaps:** Methods and classes lack proper documentation, making the code less maintainable.
3. Code review checklist:
    1. **Structure:**
        a) Does the code conform to coding standards?
        b) Is the code modular and adheres to SRP (Single Responsibility Principle)?
    2. **Documentation:**
        a) Are all comments consistent with the code?
        b) Are all public methods and classes documented?
    3. **Variables:**
        a) Are all variables properly defined with meaningful, consistent, and clear names?
        b) Is the use of magic numbers avoided?
    4. **Arithmetic Operations:**
        a) Does the code avoid comparing floating-point numbers for equality?
        b) Does the code systematically prevent rounding errors?
        c) Are divisors tested for zero or noise?
    5. **Loops and Branches:**
        a) Does every case statement have a default?
        b) Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?
    6. **Defensive Programming:**

a) Are indexes, pointers, and subscripts tested against array, record, or file bounds?
b) Are all output variables assigned?
c) Is every memory allocation de-allocated?
d) Are files checked for existence before attempting to access them?
e) Are all files and devices left in the correct state upon program termination?

7. Performance:

a) Are loops optimized for performance?
b) Are caching mechanisms in place where needed?

4. Issues found:

Source file	Class	Line	Unsatisfied Checklist Rule	Comment
SnakeHub.cs	SnakeHub	43	3 a)	Rename `pos` and `color` to more descriptive names like `snakeParts` and `snakeColor`.
SnakeHub.cs	SnakeHub	100	7 a)	Use caching for repeated operations inside the loop to improve efficiency.
SnakeHub.cs	SnakeHub	159	6 d)	Ensure `clientsStatic` is valid before using `clientsStatic.User`.
SnakeHub.cs	SnakeHub	271	6 b)	Check that `Foods` is initialized before entering the lock block.
SnakeHub.cs	SnakeHub	14	2 b)	Add comments explaining default values like `Width = 10` and `SpeedTwo = 8`.
SnakeHub.cs	SnakeHub	N/A	1 a)	Class structure follows coding standards, but names like `Sneks` and `Rng` could be more descriptive (`Snakes` and `RandomGenerator` for clarity).
SnakeHub.cs	SnakeHub	38-78	3 a)	The variable pos should be renamed to something more descriptive, such as `snakeParts`.
SnakeHub.cs	SnakeHub	38-78	6 a)	No checks for invalid or duplicate name. Consider validating input to ensure data integrity.
SnakeHub.cs	SnakeHub	30-34	2 a)	Method lacks comments explaining its purpose and behavior. Add a brief summary of its role in the observer pattern.

SnakeHub.cs	SnakeHub	97-157	6 a)	No null checks on `snek.Parts` or `Foods`. Ensure `Foods` and `Sneks` are properly initialized and not null before accessing.
SnakeHub.cs	SnakeHub	105	4 c)	Division by zero is not explicitly avoided for `Rng.Next` operations. While unlikely, validate parameters for Random methods to avoid exceptions.
SnakeHub.cs	SnakeHub	159-195	2 a)	The method lacks comments describing its behavior and the logic for determining snake collisions.
SnakeHub.cs	SnakeHub	258-260	6 e)	Ensure the singleton is disposed correctly or does not hold unused resources to prevent memory leaks.
SnakeHub.cs	SnakeHub	263-273	3 a)	The variable `foodP` could be renamed to `foodPosition` for clarity.
SnakeHub.cs	SnakeHub	197-214	6 e)	Ensure threads accessing shared resources like `Sneks` and Foods terminate gracefully to avoid deadlocks or inconsistent state.
SnakeHub.cs	SnakeHub	275-281	3 a)	Variable dir should have a more descriptive name, e.g., direction.
SnakeHub.cs	SnakeHub	275-281	2 a)	Add comments explaining how the dir value affects snake movement.
SnakeHub.cs	SnakeHub	N/A	5 a)	No switch statements are used. If used in the future, ensure all switch statements have a default case.
SnakeHub.cs	SnakeHub	N/A	6 a)	No explicit error handling for critical sections (e.g., lock blocks). Use try-catch to handle potential runtime exceptions in these blocks.
Snake.cs	Snake	30	1 b)	Break down toggling logic into smaller methods for readability.
Snake.cs	Snake	28-36	3 a)	The property Fast could be renamed to `IsBoosted` for better clarity.
Snake.cs	Snake	28-36	2 a)	The method lacks comments explaining why the movement strategy toggles between two implementations.
Snake.cs	Snake	Entire class	1 a)	Some property names (`SpeedTwo`, `Dir`) are unclear or inconsistent in naming conventions.

Snake.cs	Snake	Entire class	6 b)	The `Parts` list might have null entries if improperly initialized or manipulated.
Field.cs	Field	6-7	3 b)	Type is consistent (int), but consider using properties for better encapsulation.
Field.cs	Field	6-7	3 a)	Variable names are meaningful but could benefit from being made readonly to emphasize immutability.
Field.cs	Field	-	2 a)	The class and variables lack comments. Add descriptions for clarity, e.g., the purpose of `Width` and `Height`.
Field.cs	Field	-	6 a)	While not applicable here, if these variables will be used to define boundaries, add validations.
Food.cs	Food	8	3 b)	Using string for `Color` is functional but could lead to inconsistencies. Consider using a `Color` type.
Food.cs	Food	7	6 a)	If Position represents a point within a boundary, validations should be included to ensure it's within bounds.
Food.cs	Food	8	6 e)	Not directly applicable, but ensure consistency when dealing with color settings or rendering.

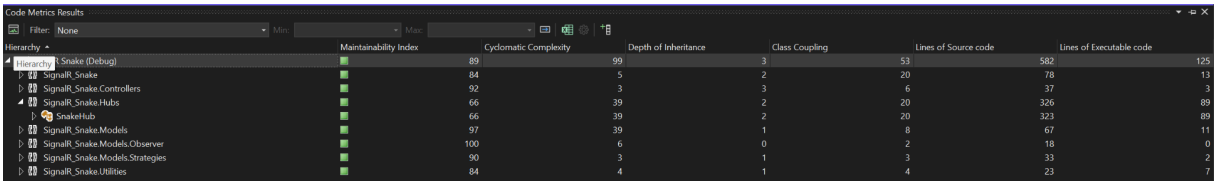
Launching static code analysis on software project

Tool Used: Microsoft Analyzer (FxCop / .NET Test)

Project: SignalR Snake

Date: [Insert Date]

The static code analysis was performed on the SignalR Snake project using the Microsoft analyzer, specifically utilizing the `RuleSet1.ruleset` configuration. A series of warnings were identified related to code design, naming conventions, performance, usage, and other aspects. This report will outline the warnings in detail, categorize them, and provide recommendations for resolving each issue.



Warnings Summary Table

Category	Warning
Design Issues	Change 'List' in 'SnakeHub.Foods' to use Collection, ReadOnlyCollection or KeyedCollection<K,V>
	Change 'List' in 'SnakeHub.observers' to use Collection, ReadOnlyCollection or KeyedCollection<K,V>
	Because field 'Field.Width' is visible outside of its declaring type, change its accessibility to private and add a property, with the same accessibility as the field has currently, to provide access to it.
	Because field 'Field.Height' is visible outside of its declaring type, change its accessibility to private and add a property, with the same accessibility as the field has currently, to provide access to it.
	Because type 'BundleConfig' contains only 'static' ('Shared' in Visual Basic) members, add a default private constructor to prevent the compiler from adding a default public constructor.
	Mark 'SignalR Snake.dll' with CLSCompliant(true) because it exposes externally visible types.
	Change 'List' in 'SnakeHub.Sneks' to use Collection, ReadOnlyCollection or KeyedCollection<K,V>
Usage Issues	Consider making 'SnakeHub.Sneks' non-public or a constant.
	Consider making 'SnakeHub.Foods' non-public or a constant.
	Consider making 'SnakeHub.Rng' non-public or a constant.
Naming Issues	Remove the underscores from namespace name 'SignalR_Snake.Controllers'.
	Correct the spelling of 'Snek' in type name 'SnekScore'.
	Remove the underscores from namespace name 'SignalR_Snake'.

	Remove the underscores from namespace name 'SignalR_Snake.Models.Strategies'.
	Remove the underscores from namespace name 'SignalR_Snake.Models.Observer'.
	Correct the spelling of 'Mvc' in type name 'MvcApplication'.
	Remove the underscores from namespace name 'SignalR_Snake.Models'.
	Correct the spelling of 'Snek' in type name 'SnekPart'.
	Remove the underscores from namespace name 'SignalR_Snake.Hubs'.
	Correct the spelling of 'Sneks' in member name 'SnakeHub.Sneks' or remove it entirely if it represents any sort of Hungarian notation.
	Correct the spelling of 'Rng' in member name 'SnakeHub.Rng' or remove it entirely if it represents any sort of Hungarian notation.
	Correct the spelling of 'Snek' in member name 'HomeController.Snek(Snake)' or remove it entirely if it represents any sort of Hungarian notation.
	Correct the spelling of 'Snek' in member name 'SnakeHub.NewSnek(string)' or remove it entirely if it represents any sort of Hungarian notation.
	Correct the spelling of 'Pos' in member name 'SnakeHub.AllPos()' or remove it entirely if it represents any sort of Hungarian notation.
	Correct the spelling of 'Dir' in member name 'SnakeHub.SendDir(double)' or remove it entirely if it represents any sort of Hungarian notation.
	Correct the spelling of 'dir' in parameter name 'dir' in method 'SnakeHub.SendDir(double)' or remove it entirely if it represents any sort of Hungarian notation.
	Correct the spelling of 'Dir' in member name 'Snake.Dir' or remove it entirely if it represents any sort of Hungarian notation.

Performance Issues	The 'this' parameter (or 'Me' in Visual Basic) of 'MvcApplication.Application_Start()' is never used. Mark the member as static (or Shared in Visual Basic) or use 'this'/'Me' in the method body or at least one property accessor, if appropriate.
	Initialize all static fields in 'SnakeHub' when those fields are declared and remove the explicit static constructor.
	The 'this' parameter (or 'Me' in Visual Basic) of 'SnakeHub.CheckCollisions()' is never used. Mark the member as static (or Shared in Visual Basic) or use 'this'/'Me' in the method body or at least one property accessor, if appropriate.
Mobility Issues	Modify the call to 'Timer.Timer(double)' in method 'SnakeHub.SnakeHub()' to set the timer interval to a value that's greater than or equal to one second.
Globalization Issues	Because the behavior of 'string.Format(string, object)' could vary based on the current user's locale settings, replace this call in 'RandomColorSingletonHelper.GenerateRandomColor()' with a call to 'string.Format(IFormatProvider, string, params object[])'.
Miscellaneous Issues	Post-build Code Analysis (FxCopCmd.exe) has been deprecated in favor of FxCop analyzers, which run during build. Refer to https://aka.ms/fxcopanalyzers to migrate to FxCop analyzers.
	Sign 'SignalR Snake.dll' with a strong name key.

The warnings highlight various code improvements across multiple areas. They emphasize better design practices, such as replacing `List` with more secure collection types like `ReadOnlyCollection`, improving encapsulation by adjusting field accessibility, and marking DLLs as `CLSCompliant`. There are also usage issues recommending making certain members non-public or constants. Naming conventions need attention, with suggestions to remove underscores from namespaces and correct terms like "Snek" and "Dir." Performance optimizations include eliminating unused parameters and adjusting timer intervals. Additionally, there are globalization concerns about ensuring locale consistency in string formatting, as well as recommendations for migrating to newer code analysis tools and signing the DLL for security.

Design and implement ONE code analysis rule

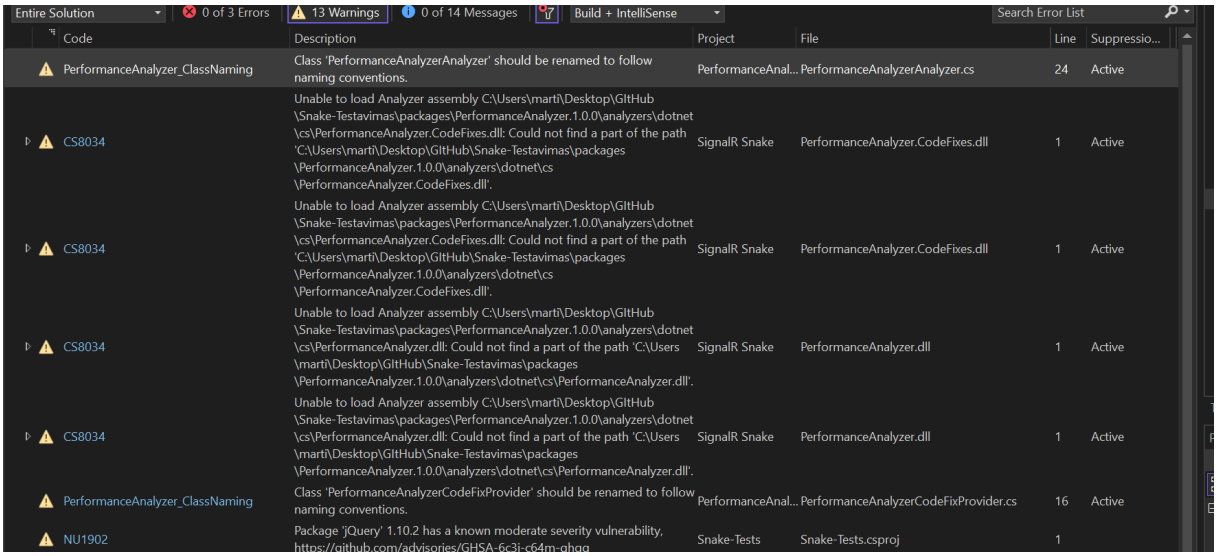
We needed to make a one new code analysis rule, so we made four to enhance software quality by detecting issues based on performance, maintainability, or naming conventions. The rules integrate into a Roslyn-based analyzer and provide actionable feedback to developers.

Steps on how we made it:

- Create an Analyzer class and define the rules that we want to enforce(in our case uncached Count calls, ensuring proper naming convention).
- Create a name for the rule and decide on the severity level of the rule.
- Use the DiagnosticAnalyzer class to define the rule's logic and CodeFixProvider for auto-fix capabilities (probably will work in the future).
- Test the set rules with example code snippets.
- Pack and publish as a NuGet package.
- Finally, test the rules with our project.

After testing the new rules our code had some warnings from the new testing rules:

PerformanceAnalyzer_ClassNaming	Class 'BundleConfigTests' should be renamed to follow naming conventions.	Snake-Tests	BundleConfigTests.cs	9	Active
PerformanceAnalyzer_ClassNaming	Class 'FieldTests' should be renamed to follow naming conventions.	Snake-Tests	FieldTests.cs	7	Active
PerformanceAnalyzer_ClassNaming	Class 'FilterConfigTests' should be renamed to follow naming conventions.	Snake-Tests	FilterConfigTests.cs	9	Active
PerformanceAnalyzer_ClassNaming	Class 'FoodTests' should be renamed to follow naming conventions.	Snake-Tests	FoodTests.cs	8	Active
PerformanceAnalyzer_ClassNaming	Class 'HomeControllerTests' should be renamed to follow naming conventions.	Snake-Tests	HomeControllerTests.cs	8	Active
PerformanceAnalyzer_ClassNaming	Class 'HomeController' should be renamed to follow naming conventions.	Snake-Tests	HomeControllerTests.cs	33	Active
PerformanceAnalyzer_ClassNaming	Class 'SnakeHubIntegrationTests' should be renamed to follow naming conventions.	Snake-Tests	IntegrationTests.cs	19	Active
PerformanceAnalyzer_ClassNaming	Class 'MovementStrategyTests' should be renamed to follow naming conventions.	Snake-Tests	MovementStrategyTests.cs	8	Active
PerformanceAnalyzer_ClassNaming	Class 'ObserverTests' should be renamed to follow naming conventions.	Snake-Tests	ObserverTests.cs	8	Active
PerformanceAnalyzer_ClassNaming	Class 'SnakeHub' should be renamed to follow naming conventions.	Snake-Tests	ObserverTests.cs	13	Active
PerformanceAnalyzer_ClassNaming	Class 'SnakeObserver' should be renamed to follow naming conventions.	Snake-Tests	ObserverTests.cs	45	Active



Summary

The laboratory work focused on reviewing the SignalR Snake 2D game's code and applying static code analysis to enhance its quality. Key objectives included identifying potential issues, correcting them, and developing custom static analysis rules for improved maintainability and performance.