Turiba University

Ekhlas Rayees Khan

FRIENDLY CODING

PROFESSIONAL BACHELOR THESIS

Study Programme: Computer Systems

Author:	**Ekhlas Rayees Khan**
***Thesis* Advisor:**	**Janis Peksa**

Riga, 2025

Introduction

Nowadays, the quality of code plays a crucial role in the overall success of any project. This leads to the concept of "friendly coding," which simply refers to writing code that is easy to read and understand. The main idea is to make codes clear for other developers, so that they can also understand what the author wanted to explain

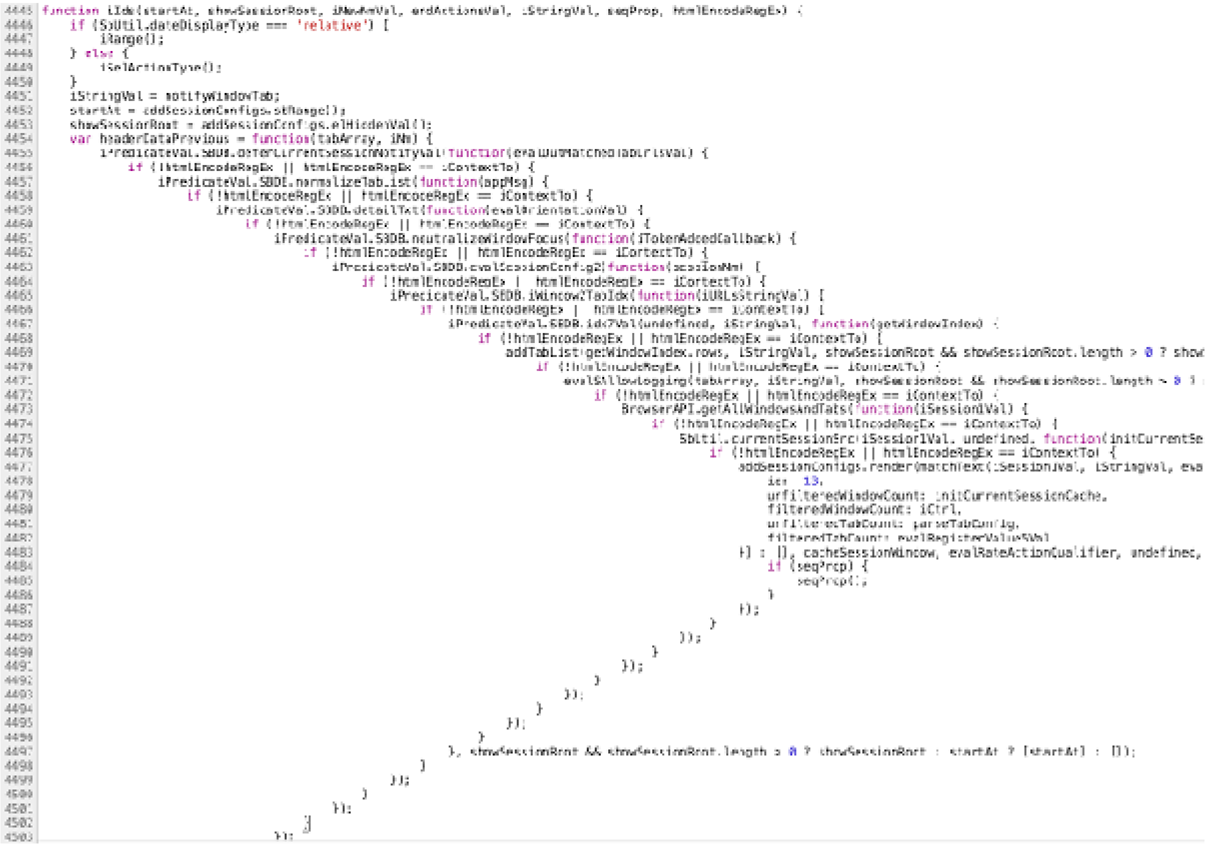
What is Friendly Coding?

At its core, friendly coding refers to the practice of writing software in a clear, organized, and human-friendly way. The goal of using friendly coding is to ensure that the code is understandable and that it avoids unnecessary complexity

This practice is built on a few key points.

The main point of friendly coding is **Readability**: the code should be easy to understand for a developer. This is supported by **Simplicity**, which helps the author to avoid over-complicated solutions. Finally, **Documentation** is important, allowing authors to write comments on the place where it is necessary.

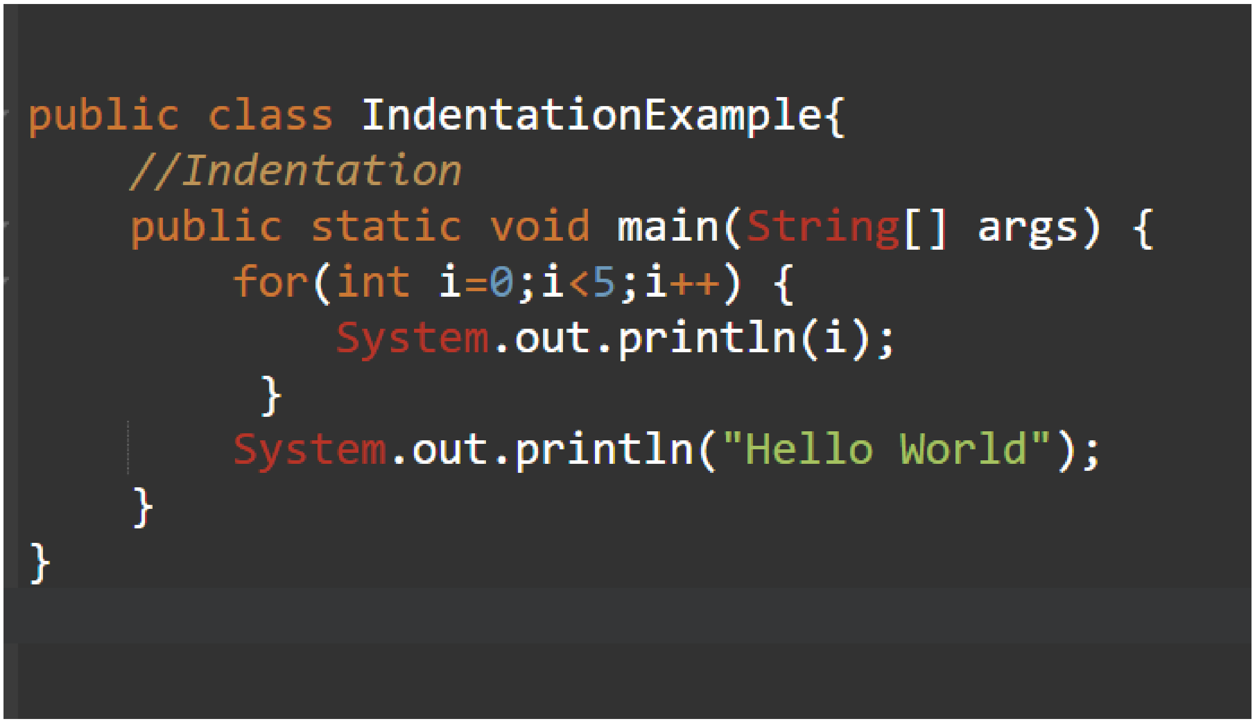
To see the difference, consider this "unfriendly" code:



```
function iIdx(startAt, showSessionRoot, iNewAmVal, erdActionsVal, iStringVal, seqProp, htmlEncodeRegEx) {
    if (SbUtil.dateDisplayType === 'relative') {
        iRange();
    } else {
        iSelActionType();
    }
    iStringVal = notifyWindowTab;
    startAt = addSessionConfigs.sbRange();
    showSessionRoot = addSessionConfigs.elHiddenVal();
    var headerDataPrevious = function(tabArray, iNm) {
        iPredicateVal.SBDB.deferCurrentSessionNotifyVal(function(evalOutMatchedTabCritsVal) {
            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                iPredicateVal.SBDB.normalizeTabList(function(appMsg) {
                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                        iPredicateVal.SBDB.detailTxt(function(evalOrientationVal) {
                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                iPredicateVal.SBDB.neutralizeWindowFocus(function(iTokenAddedCallback) {
                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                        iPredicateVal.SBDB.evalSessionConfig2(function(sessionNm) {
                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                iPredicateVal.SBDB.iWindow2TabIdx(function(iURLsStringVal) {
                                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                        iPredicateVal.SBDB.idx7Val(undefined, iStringVal, function(getWindowIndex) {
                                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                addTabList(getWindowIndex.rows, iStringVal, showSessionRoot && showSessionRoot.length > 0 ? show
                                                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                        evalSAllowLogging(tabArray, iStringVal, showSessionRoot && showSessionRoot.length > 0 ? :
                                                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                                BrowserAPI.getAllWindowsAndTabs(function(iSessionIVal) {
                                                                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                                        SbUtil.currentSessionSrc(iSessionIVal, undefined, function(initCurrentSe
                                                                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                                                addSessionConfigs.render(matchText(iSessionIVal, iStringVal, eva
                                                                                                    id: 13,
                                                                                                    unfilteredWindowCount: initCurrentSessionCache,
                                                                                                    filteredWindowCount: iCtrl,
                                                                                                    unfilteredTabCount: parseTabConfig,
                                                                                                    filteredTabCount: evalRegisterValue5Val
                                                                                                }] : [], cacheSessionWindow, evalRateActionQualifier, undefined,
                                                                                                    if (seqProp) {
                                                                                                        seqProp();
                                                                                                    }
                                                                                                });
                                                                                            }
                                                                                        });
                                                                                    }
                                                                                });
                                                                            }
                                                                        });
                                                                    }
                                                                });
                                                            }
                                                        }, showSessionRoot && showSessionRoot.length > 0 ? showSessionRoot : startAt ? [startAt] : []);
                                                    }
                                                });
                                            }
                                        });
                                    }
                                });
```

Friendly Code

This "friendly" version is functionally identical but far more readable.



```
public class IndentationExample{
    //Indentation
    public static void main(String[] args) {
        for(int i=0;i<5;i++) {
            System.out.println(i);
         }
        System.out.println("Hello World");
    }
}
```

Benefits of Friendly Coding

- Easier **Bug and Error Detection:** Clean code makes it easier to find bugs and errors.
- Faster **Introduction for New Employees:** New team members can understand the codebase more quickly.
- Improving **Collaboration with Colleagues:** Readability is key to better teamwork and collaboration.
- Reduced **Maintenance Costs:** The practice leads to decreased time and expense for servicing the code

Conclusion

Friendly coding is a fundamental practice that enhances teamwork, efficiency, and long-term software quality. By using clean coding, developers contribute to projects that are more manageable, scalable, and reliable. Whether in academic assignments or professional environments, friendly coding remains an essential skill.