**Clean Codes**

**What is Bad Code?**

Bad Code ဆိုတာ readability, maintenance, business logic ခက်ခဲတဲ့ ကုဒ်တွေဖြစ်တယ်။ Clean Codes ဆိုတာ အဲ့လိုပြဿနာတွေ ဖြစ်မနေအောင်ရေးထားတဲ့ကုဒ်ကို ခေါ်တယ်။ Good Code ဖြစ်ဖို့ဆိုရင် mindset ကောင်းကောင်းရှိရမယ် ကိုယ်လုပ်တဲ့အလုပ်နဲ့ ရေးလိုက်တဲ့အရာတွေပေါ် ကောင်းမွန်ဖို့ စိတ်ထားနဲ့ qualityကောင်းဖို့ စဥ်းစားပီး ရေးသင့်တယ်။ ရေးလိုက်တဲ့ ကုဒ်က အလုပ်လုပ်ယုံတင်မကပဲ ကျန်တဲ့ maintainability, performance တွေပါ စဥ်းစားပီး professional ကျကျရေးသင့်တယ်။

**What is Clean Code?**

Clean Codeဆိုတာ ဖတ်လို့လွယ်ရမယ်, maintenance လုပ်ဖို့လွယ်ရမယ်, ပြင်လို့လွယ်ရမယ် အဲ့လိုကုဒ်မျိုးကို clean codeလို့ ခေါ်တယ်။

**The Boy Scout Rule**

Boy Scout Ruleဆိုတာ တောင်တွေ တောတေွမှာ campချရင် ‌ေရာက်ခဲ့တဲ့နေရာကို အရင်ထက် ပိုသန့်ရှင်းအောင် ထားသလိုပဲ ကိုယ်ရောက်ခဲ့တဲ့ အလုပ်မှာ ကုဒ်တွေကို ရှုပ်နေရင် အရင်ကထက်ပိုရှင်းအောင် ပြင်ဆင်လုပ်ပေးခဲ့ရမယ်။

**Broken Window Theory**

Broken Window Theory ဆိုတာ ကားတစီးက ပျက်ပီး မှန်ကွဲနေရင် အထဲ၀င်ဖို့ မလွယ်ရင် နောက်ထပ်မှန်ကို ထပ်ခွဲပီး ၀င်ဖို့ ၀င်မလေးသလိုပဲ ကုဒ်base တခုမှာ ညစ်ပတ်ပီး refactor မလုပ်ရင် ကြာရင် ကုဒ်တွေ ရှုပ်ပွပီး ဟိုလူပြင် ဒီလူပြင်နဲ့ ထပ်ရှုပ်တာမျိုး မဖြစ်စေနဲ့ ၊ ကိုယ်ထိလိုက်တဲ့ ကုဒ်ဆိုရင် ညစ်ပတ်နေရင်တောင် အချိန်ယူပီး refactoring လုပ်သင့်ပါတယ်။

1. **Meaning Full Name**

Naming တွေပေးရင် သတ်မှတ်ထားတဲ့ language ရဲ့ coding standard ကို လိုက်နာသင့်တယ်၊

* 1. **Use Indentation Reveal Names**

ဒါကဘာကိုပြောချင်တာလဲ ဆိုရင် variable namesတွေပေးရင် single letter တွေမဖြစ်သင့်ဘူး looping index and algorithm တွေကလွဲရင် မပေးသင့်ဘူး ။

* 1. **Must Be Readability Code**

ကိုယ့်ကုဒ်ကို တခါမှ မမြင်ဘူးတဲ့သူကနေ ဖတ်ရင် နားလည်အောင် အချိန်ဘယ်လောက်ကြာလဲ ဆိုတာ နည်းနိုင်သမျှနည်းရမယ် ကိုယ့်ကုဒ်ကို ဘယ်သူဖတ်ဖတ် နားလည်ဖို့ လွယ်ကူရမယ်၊

* 1. **Avoid Disinformation**

နားလည်မှူလွဲစေမယ့် နာမည်ပေးမှုမျိုးကို ရှောင်ကျဥ်ပါ

* 1. **Use Only One Ref or one variable for one entity.**

Variable reference ကို တက်နိုင်ရင် တခုထဲပဲဖြစ်ရမယ်၊ state ကိုပြောင်းလဲစေမယ့်ကောင်မျိုး နေရာ အများကြီးမဖြစ်ရဘူး။

Eg.

let a = {};

let b = a;

processB(b);

အဲ့လိုမျိုး variable reference များစေတဲ့ကုဒ်မျိုးကို ရှောင်ကျဥ်ရမယ်။

* 1. **Must be low cognitive load in Method**

Method တခုကို နားလည်ဖို့ သူအထဲမှာ ထပ်ခေါ်ထားတဲ့ function တွေ method တွေကို ပြန်ပီး နားလည်အောင်လုပ်ဖို့ လိုတာမျိုးမဖြစ်ရဘူး ။ ခေါ်ထားတဲ့ နာမည်ကို ကြည့်လိုက်တာနဲ့ ဘာလုပ်ချင်တာလဲဆိုတာကို သိရမယ်။ low connection method or function is better.

* 1. **Use Pronounceable Names**

Name ကို အသံထွက်ဖတ်ရင် နားလည်နိုင်တာမျိုးပဲ ပေးရမယ်။

Eg.

class DtaStre123{} အဲ့လိုမပေးပဲ

class Customer{} အဲ့လို မျိုးပေးထားတာဖြစ်ရမယ်။

* 1. **Must be Searchable Name**

Data ပြန်ရှာတဲ့ အခါမှာ ခဲယဥ်းတဲ့ aa, ab, အဲ့လို နာမည်တွေအစား userNameအဲ့လို ရှာဖို့ လွယ်တဲ့ နာမည်မျိုးပေးရယ်။

* 1. **Avoid Mental Mapping**

သုံးလိုက်တဲ့ Method, variable တခုချင်းဆီက ဘာလုပ်တယ်ဆိုတာမျိုး စိတ်ကနေမှတ်ပီး ပြန်လုပ်နေရတယ့် naming convention မျိုးရှောင်ကျဥ်ရမယ်။

Eg. if (errorCode == 1022) // Invalid Amount

ဆိုတာမျိုးကိုရှောင်ကျဥ်ရမယ် အဲ့အစား const, enum သုံးတာမျိုး နဲ့ ပြန် refactorလုပ်သင့်တယ်

Eg. const INVALID\_AMOUNT = 1022;

If (errorCode == INVALID\_AMOUNT)

အဲ့လိုမျိုးအစားထိုး သုံးနိုင်တယ်

* 1. **Class Name must be Noun and Methods must be Verb or VerbPhrase.**
  2. **Pick One Word Per Concept**

ရေးနေတဲ့ Project မှာ naming ကို အမျိုးမျိုးပြောင်းမသုံးသင့်ဘူး စံတခုပဲ သတ်မှတ်ပီးသုံးသင့်တယ် ဥပမာနေနဲ့ ဒေဝာာဆွဲထုတ်တာမှာ တခုကို fetch, နောက်တခုကျ reterieve, နောက်တခုကျ get အဲ့လိုမျိုးမသုံးပဲ သုံးမယ့်method တိုင်းကို getဆိုget fetchဆိုfetch နဲ့သုံး သင့်တယ်

**Functions**

**Why Use Functions?**

Function ကို reusable, abstraction တွက်သုံးတယ်၊ abstraction ဆိုတာ low level details process ကိုသိစရာမလိုပဲ နားလည်လွယ်တဲ့ simple one နဲ့ အစားထိုးလိုက်တာကိုခေါ်တယ်။ ဥပမာနေနဲ့ဆို ကားဆိုရင် ကားအထဲက engine တွေ ၀ါယာတွေ ဘရိတ်တွေ နောက်ကွယ်က ဘယ်လိုလုပ်နေတာလဲဆိုတာ သိဖို့မလိုပဲ ကားမောင်းမယ်ဆို ရပ်ရင် ဘရိတ်အုပ်မယ် မောင်းမယ်ဆို အင်ဂျင်စမယ် အဲ့လို complexity တွေကို နားလည်လွဘ်အောင် အခြားတခုနဲ့ အစားထိုးထားတမျိုးကိုခေါ်တယ်၊

**Function Must Be Small**

Function 1ခုက small ဖြစ်ရမယ်၊ small ဖြစ်တယ်ဆိုတဲ့ နေရာမှာ SRP ကို ပြေလည်ရယ် ပြောချင်တာက လုပ်မယ့် responsibility တခုပဲရှိရမယ်။ Scroll ဆွဲပီး ကြည့်နေရတာမျိုးထိ ရှည်နေတာမျိုး မဖြစ်သင့်ဘူး၊

**Must Be SRP.**

SRP(single Responsibility pattern) ကိုပြေလည်ရမယ်။ function 1ခုဟာ process 1ခုကိုပဲ လုပ်ဆောင်ရမယ်။ ဥပမာနေနဲ့ဆို total money တွက်တဲ့ function မှာ average person တို့ average moneyတို့ တွက်တာမျိုးမပါရဘူး

**Use Block and Indentation**

သက်ဆိုင်ရာ လုပ်ဆောင်ချက်တူရာတခုပီးတိုင်း empty lineလိုမျိုးနဲ့ block ခွဲသင့်တယ်။ if else လိုမျိုး logical expressionတွေပီးတိုင်းမှာ block ခွဲသင့်တယ်။

**Function Must Be From High Level Abstraction To Low Level Abstraction**

Abstrationတွေက high ကနေ low သွားတာမျိုးပဲ ဖြစ်ရမယ်။ ဥပမာနေနဲ့ဆို array ထဲကနေ ပထမ သုံးခုကို ယူမယ်ဆိုတဲ့ function မှာဆိုရင် ။

function takeTopThree(){

sort();

takeTopThree();

}

function sort(){}

function takeTopThree(){};

အဲ့လိုမျိုး abstraction က top to low ဖြစ်ရမယ်။ အောက်ကလိုမျိုး low to top မဖြစ်ရဘူး။

function sort(){}

function takeTopThree(){};

function takeTopThree(){

sort();

takeTopThree();

}

**Use Descriptive Naming**

Function မှာ နာမည်ပေးရင် verb form ကို ပေးပီး နာမည်ကို ကြည့်တာနဲ့ ဘာလုပ်ချင်လဲဆိုတာ သိရမယ် ပေးထားတဲ့ နာမည်မှာ And Orပါတာမျိုးတွေမဖြစ်သင့်ဘူး

**Principle of Least Astonishment**

Function တွေက principle of least astonishmentကိုလိုက်နာရမယ် ပြောချင်တာက get ဆိုတဲ့ function မှာ setရဲ့ လုပ်ဆောင်ချက်မျိုးတွေမရှိသင့်ဘူး

**Function Arguments should be 3 Most**

Function တွေမှာ ပေးလိုက်တဲ့ argument အ‌ေရတွက်ဟာ အများဆုံး ၃ခုပဲဖြစ်သင့်တယ် သုံးခုထက်ကျော််ရင် SRP ကိုချိုးဖောက်သလိုဖြစ်တယ် အဲ့တော့ သုံးခုထက်ပိုတာကို ပို့မယ်ဆိုရင် DTO လိုမျိုး သက်ဆိုင်ရာ model classမျိုးဆောက်ပေးတာမျိုးလုပ်သင့်တယ်

**Function Should only Query Or Command.**

Function 1ခုက ဒေတာ set လုပ်တာ command or data ဆွဲထုတ်တာ query တခုပဲ ဖြစ်သင့်တယ် ၂ခုလုံးကို ရောလုပ်ထားတာမျိုး မဖြစ်သင့်ဘူး

**Avoid Passing Flag Arguments**

State တွေကို handle လုပ်ဖို့တွက်နဲ့ Boolean arguments တွေပေးလိုက်တာမျိုးမလုပ်သင့်ဘူး ဘာလို့လဲဆိုတော့ Boolean value pass လုပ်လိုက်ပီဆိုတာနဲ့ conditional logic တွေစစ်ဖို့လိုလာပီ အဲ့တော့ အဲ့ကောင်က SRP ကိုချိုးဖောက်တယ် ဒါမို့ မပေးသင့်ဘူး

**Set Different Parameters Type In Argument List**

Function 1ခုမှာ တူညီတဲ့ ဒေတာtypeတွေ ဆက်တိုက်ဖြစ်နေပီး မတူတာလဲ ပါရင် တူတဲ့ type တွေကြားမှာ မတူတဲ့type ကိုထားသင့်တယ် ဘာလို့လဲဆိုတူတဲ့ ‌ေဒတာtype ဆို accidently wrong typing အရ မှားနိုင်ခြေများလို့ပါ ဥပမာနေနဲ့ဆို

function area(int width, int height, long radius){} အစား

function area(int width, long radius, int height){} အဲ့လိုမျိုး ဖြစ်နိုင်ရင် သုံးသင့်တယ်။

**Easy To Use Hard To Misuse (EUHM)**

Method 1 ခုကို ခေါ်ဖို့ အခြား method တွေခေါ်ရမယ်ဆိုရင် EUHM principle ကို ချိုးဖောက်တာပဲ အဲ့တော့ အဲ့လိုမျိုးခေါ်ဖို့လိုလာပီဆို တိုက်ရိုက်ကြားထဲမှာ အကုန်ခေါ်ထားပေးတဲ့ကောင်မျိုးနေနဲ့ ရှိသင့်တယ် ဥပမာနေနဲ့ဆို api က onNext ခေါ်ပီး onCompletဆိုပီး ပီးမပီး စစ်ဖို့ခေါ်နေရတယ်ဆို User ကရေးရင်းမေ့ကျန်နေခဲ့တာမျိုး ဖြစ်နိုင်တယ် အဲ့တော့ အဲ့လို design အစား onNext ခေါ်လိုက်တာနဲ့ onComplete လားဆိုတာ checkပီးသား ဖြစ်ရင် ပိုကောင်းတယ်

**Should No Side Effect**

Function 1ခုက သူ့ကိုပေးလိုက်တဲ့ parameter အပြင်အခြားဟာတွေကို သုံးနေပီဆိုရင် side Effect ရှီနေပီ အဲ့လိုမျိုးကို အတက်နိုင်ဆုံး ရှောင်ရှားသင့်တယ် ဥပမာ user Login password check တဲ့ function မှာဆို ပေးလိုက်တဲ့ password တင်မက အခြား session တွေ ပါ check ပီး external data တွေ ပေါ် effect သတ်ရောက်နေပီဆို အဲ့တာ side effect ဖြစ်တာလို့ပြောလို့ရတယ်

**Passing Parameter Limit Scope**

Function ကိုပို့ပေးလိုက်တဲ့ parameter ကို အဲ့သုံးနေတဲ့ function ကနေနောက်တခုကို ထပ်ပေးတဲ့ scope က အများဆုံး ၂ခုပဲ ဖြစ်သင့်တယ် အဲ့ထက်များသွားရင် code base များလာတာနဲ့အမျှ handle လုပ်ရတာခက်သွားမယ်၊

**Command Query Separation**

Command ဆိုတာ method ကို process လုပ်ခိုင်းတာမျိုး Query ဆိုတာကကျတော့ getter setter သတ်မှတ်တာမျိုး ပြောချင်တာက method တွေမှာ processလုပ်တာသတ်သတ် getter, setterလုပ်တာ သပ်သပ်ရေးရမယ်ပြောတာ responsibilityတွေ ရောမရေးသင့်ဘူး

**Avoid Generic Exception**

Common Exception ဖြစ်တဲ့ catch(Exception ex)ဆိုတာမျိုး ကိုရှောင်သင့်တယ် သူဖြစ်တဲ့ exception ကို details အလိုက်ဖမ်းသင့်တယ် ဥပမာ DbException, SqlException အဲ့လို တခုချင်းစီ အလိုက်သေခြာရေးသင့်တယ်

**Don’t repeat yourself (DRY)**

3ခုအထပ်ပိုပီး ထပ်တလဲလဲသုံးနေရတာမျိုးဆိုရင် function တခု ထပ်ခွဲပီးသုံးပါ

**Structure Programming Vs Unstructured Programming**

Structure Programming ဆိုတာ သူတို့ရဲ့ loop တွေဟာ Single Entry Single Exist ဖြစ်ရမယ် ဆိုလိုချင်တာက loop ထဲ၀င်လာတဲ့နေရာကလဲ တနေရာ ပြန်ထွက်မယ်ဆိုလဲ အဲ့နေရာကဖြစ်ရမယ်

External side effect တွေအရ exist change သွားတာမျိုးမဖြစ်ရဘူး

Unstructured Programmingဆိုတာကျ အဲ့လိုမဟုတ်ပဲ exist point က အမျိုးမျိုးဖြစ်နိုင်တယ် ဥပမာ နေနဲ့ဆိုရင် အောက်က go program ကိုကြည့်ပါ

import fmt;

function main(){

i := 0;

loop:

fmt.println(‘loop start’);

goto end

i += 1;

if(i < 10){

goto loop

}

end :

Fmt.println(‘end’);

}

ဒိprogram မှာဆို entry က loop ကနေ ဖြစ်ပေမယ့် exist က end ကို jump လုပ်ပီး ဖြစ်နိုင်သလို if statement ပီးမှလည်းဖြစ်နိုင်တယ် အဲ့လိုမျိုး exist အများကြီးဖြစ်န်ုင်တဲ့ကောင်မျိုးကို unstructured programming လို့ခေါ်တယ်၊

Break, continue, returnတွေအရမ်းသုံးရင်လည်း unstructured programming ဖြစ်စေတယ်၊ အဲ့ကောင်တွေအစား logical condition တွေ if elseတွေနဲ့ သုံးနိုင်တယ် ဒါပေမယ့် break, return သုံးလို့ အဆင်ပြေတဲ့ကောင်မျိုးဆိုရင်တော့ သုံးလိုက်လို့ရတယ် အဓိကက over used မဖြစ်စေဖို့ပါပဲ

**Comments**

Code တွေမှာ comment တွေနေရာတိုင်းလိုက်ရေးတဲ့ အစား ကြည့်လိုက်တာနဲ့ ဘာလုပ်ချင်တာလဲ ဆိုတာရှင်းလင်းပီး clean code ဖြစ်နေတာမျိုး ဖြစ်သင့်တယ်။ Try first for good code. Documentationလိုမျိုး လိုအပ်မှပဲ comment ‌ေရးသင့်ပါတယ်။

**Should Write Legal Comment**

ဘယ်သူက ဘယ်company ကဆိုတဲ့ legal comment မျိုးတော့ရေးသင့်တယ် ဥပမာနေနဲ့

// Copy Right From Some one company

**Informative Comments**

Api တို့ Documentationတို့လိုမျိုးတွေမှာ ဒီmethod ကဘာလုပ်တယ် ဘာဒေတာ ပြန်ပေးတယ် စတာမျိုးတွက်ဆိုရင်သုံးတယ် ဥပမာနေနဲ့

// Respond an instance of responder

Protected abstract Respond RespondTest();

**Warning Of Consequences**

Codeမှာ ဒီကောင်ကဘာလို့ရေးထားတယ် ဘယ်လိုအကျိုးရှိလို့ရေးထားတယ် ပြင်လိုက်ရင် ဘယ်လိုဖြစ်သွာနိုင်တယ်ဆိုတာမျိုးရေးထားတယ့်ကောင်ကိုပြောတာ

**TODO Comments**

ဘာလုပ်ဖို့ကျန်သေးတယ် ဆိုတာမျိုးရေးတဲ့ကောင်

**Bad Comments**

Bad Comments ဆိုတာ သိရဲ့သားနဲ့ မလိုပဲ ထပ်ထပ်ရေးနေတဲ့ comment မျိုးကိုခေါ်တယ်

**Avoid Misleading comments**

ရေးထားတဲ့ ကုဒ်ကို နားလည်မှုလွဲမှားစေတဲ့ မမှန်ကန်တဲ့comment မျိုးကိုခေါ်တယ်

**Avoid Journal comment**

Journal commentဆိုတာ version controlလိုမျိုး ဘာတွေလုပ်ခဲ့တယ်ဆိုတာမျိုးကို ရေးထားတဲ့ commentမျိုးကိုခေါ်တယ် အဲ့အစား version control systemတွေကို သုံးသင့်တယ်

**Avoid Comment-Out Code**

မသုံးတော့လို comment ပိတ်ထားတဲ့ codeတွေကို ဖြတ်သင့်တယ်

**Avoid Inobvious comments** - ဘာမှမသတ်ဆိုင်တဲ့ comment မျိုး