C# Coding Standards

C# Coding Standards
အခြေခံသိကောင်းစရာများ

Myanmar Language

Author : Zawminsoel

aspcrazy92@gmail.com

အားလုံးဘဲ မင်္ဂလာပါခင်ဗျာ...။☺☺☺

ကျွန်တော်တို့ဒီတစ်ခေါက်လေ့လာကြမှာကတော့ C# , ASP.Net Coding Standard အကြောင်းဘဲဖြစ်ပါတယ်

ဘာကြောင့် ဒီစာအုပ်ကိုရေးသားရလဲဆိုတော့ Coding တွေရေးမဲ့ Programmer

အသစ်တွေအတွက် စနစ်တကျဖြစ်စေချင်လို့ပါ။

Professional တွေကတော့ ဒီစာအုပ်ကို ဖတ်စရာမလိုပါဘူး။

ဘာဖြစ်လို့လဲဆိုတော့ သူတို့က ကျွန်တော့်ထက် အတွေ့အကြုံအများကြီးရှိတဲ့အတွက် မလိုဘူးလို့ထင်ပါတယ်။

C# Language ကိုလေ့လာမဲ့သူတစ်ယောက်ဆိုရင်တော့ ဒီစာအုပ်ကိုဖတ်သင့်တယ်လို့ကျွန်တော်ထင်ပါတယ်။

ဒါကြောင့်လည်း Programming မှာ C# Language အတွက် Coding Standard

တွေကိုရေးထားတာဖြစ်ပါတယ်။

တစ်ခုတော့ရှိပါတယ်။လုံးဝ Beginner ဆိုရင်တော့ ဒီစာအုပ်ထဲမှာပါတဲ့ Syntax တွေ..Logic တွေနဲ့ စကားလုံး

တွေကိုတော့ နားလည်မှာမဟုတ်ပါဘူး။

ဒီစာအုပ်အထဲမှာပါတဲ့အကြောင်းအရာတွေကတော့ C# Language ရေးသားတဲ့အခါမှာ လိုက်နာသင့်တဲ့အ

ချက်လေးတွေကို စုစည်းဖော်ပြပေးထားပါတယ်။

English လိုဖတ်ရမှာအခက်အခဲရှိသူတွေ အတွက်တော့ အဆင်ပြေမယ်လို့မျှော်လင့်ပါတယ်။ဒါကြောင့် ဘာသာ

ပြန်သင့်တာပြန်....တစ်ချို့အရာတွေကို ကိုယ်တွေ့အတွေ့အကြုံလေးတွေကိုလည်းထည့်ပေးထားပါတယ်။

 စာအရေးအသား အမှားများပါဝင်ခဲ့ရင်လည်း စိတ်မရှိပါနဲ့လို့တောင်းဆိုချင်ပါတယ်။

ကျွန်တော်သည်လည်း လေ့လာနေဆဲလူသားတစ်ယောက်ဖြစ်တာကြောင့် အမှားနဲ့မကင်းပါဘူးခင်ဗျာ...။

တစ်ချို့ စာသားများသည်လည်း မည်သူ့ ကိုမှ ရည်ရွယ်ခြင်းမဟုတ်သောကြောင့် တိုက်ဆိုင်ခဲ့ပါက
မရည်ရွယ်ပါကြောင်း အသိပေးပါတယ်ခင်ဗျာ..။

ဒါဆိုရင် ကျွန်တော်တို့ ဘာကြောင့် Coding Standard ကိုလိုက်နာရသလဲဆိုတာကို

လေ့လာကြည့်လိုက်ရအောင်။

C# Coding Standard

လူတိုင်း Coding ရေးနိုင်တယ်။ Coding ရေးတဲ့အတွေ့အကြုံနဲနဲရှိတာနဲ့ Coding ရေးနိုင်ကြပါတယ်။

ဒါပေမဲ့အချို့လူတွေက ရေးကောင်းတိုင်းရေးနေကြတယ်။Coding တွေအများကြီးရေးနေရုံနဲ့လည်း ကျွမ်းကျင်တဲ့ Programmer မဖြစ်သေးပါဘူး။ဘယ်လောက်ဘဲ Logic ကောင်းကောင်း ဘယ်လောက်ဘဲတီထွင်နိုင်စွမ်းရှိရှိ ဘယ်လောက်ဘဲ Software တွေအများကြီးရေးခဲ့ပါစေ…နည်းလမ်းမမှန်တဲ့ Coding ရေးနည်းနဲ့သာဆိုရင်သူရေးထားတဲ့ App က အဆင့်အတန်းမှီတဲ့ App တစ်ခုဖြစ်မှာမဟုတ်ပါဘူး။

တစ်ချို့ဆိုရင် Program တွေကိုတော့ရေးနိုင်ပါတယ်။ဒါပေမဲ့သူတို့ရေးထားတဲ့ Programming Code တွေကိုသွားကြည့်လိုက်ရင်တော့ အရင်ခေတ် Visual Basic နဲ့မခြား Button ကို Double Click ခေါက်ပြီးတော့ အဲ့ဒီအထဲမှာဘဲ ကိုယ်လုပ်ချင်တဲ့အလုပ်တွေကို သိမ်းကျုံးပြီးလုပ်ဆောင်နေကြတာကိုလည်းတွေ့ရပါတယ်။

Tier တွေအကြောင်းလဲစနစ်တကျမသိ…OOP Rules တွေကိုလည်းမသိ…Layer တွေခွဲရေးရမှန်းလည်း

မသိ…ကိုယ်ကစေတနာနဲ့သွားပြောပြန်လည်း မကောင်းထင်ခံရပြန်ရော…။

ဒါကြောင့် နိုင်ငံတကာကို အနည်းနဲ့အများ တော့ရင်ဘောင်တန်းနိုင်အောင် ကျွန်တော်တို့မြန်မာလူမျိုးတွေလည်းစနစ်တကျလေ့လာသင့်တယ်လို့မြင်ပါတယ်။

ဒါဆိုရင် အဆင့်အတန်းမှီတဲ့ App ဖြစ်ချင်တယ်ဆိုရင်တော့ မှန်ကန်တဲ့ Coding ရေးနည်းကိုကျင့်သုံးရမှာဖြစ်ပါတယ်။

ဒါဆိုရင်မေးစရာရှိလာပါပြီ။ကျွန်တော့်ကို…" မင်းကကော လိုက်နာလို့လား" ဆိုပြီးတော့မေးစရာရှိလာပါပြီ။

ဟုတ်ပါတယ်။ကျွန်တော်လည်း အရင်တုန်းက Coding Standard အကြောင်းမသိသေးတဲ့အတွက်ကြောင့်

မလိုက်နာဘဲနဲ့ရေးခဲ့တာတွေအများကြီးပါ။ဒါပေမဲ့ အမှားဆိုတဲ့အရာကို ပြင်ရမယ်ဆိုတဲ့ အသိနဲ့ Coding Standard အကြောင်းကိုလေ့လာပြီးတဲ့အခါမှာတော့ Coding Standard ကိုလိုက်နာပြီးတော့

ရေးသားနေပါတယ်။

ဒါကြောင့် စလေ့လာ ကတည်းက စနစ်တကျ လေ့လာခဲ့ရင် အကောင်းဆုံးလို့ယူဆတဲ့အတွက်ကြောင့် ဒီစာအုပ်လေးကိုရေးသားပြုစုထားတာဖြစ်ပါတယ်။

ဒါဆိုရင်မှန်ကန်တဲ့ Coding ရေးနည်းဆိုတာဘာလဲ။မှန်ကန်တဲ့ Coding ရေးနည်းဆိုတာကတော့ Coding

ရေးနေတဲ့လူတိုင်းကိုလိုက်မေးရင်တော့ အဖြေကတစ်မျိုးနဲ့တစ်မျိုးမတူအောင် ထွက်လာမှာဖြစ်ပါတယ်။

မှန်ကန်တဲ့ Coding ရေးနည်းဆိုတာကတော့ Coding ရေးတဲ့အခါမှ စနစ်တကျနဲ့ ချမှတ်ထားတဲ့ Rules တွေကို လိုက်နာပြီးတော့ရေးသားခြင်းကိုဆိုလိုပါတယ်။

နိုင်ငံတကာ Programmer တွေ…Programming Language တွေကိုတီထွင်သူတွေက သူတို့ Language ကို သုံးပြီးတော့ရေးမယ်ဆိုရင် ဒီအချက်လေးတွေကိုလိုက်နာသင့်တယ်ဆိုပြီးတော့ Coding Standard အချို့ကိုပြဋ္ဌာန်းထားတာဖြစ်ပါတယ်။

မှန်မှန်ကန်ကန်နဲ့စနစ်တကျရေးချင်တယ်ဆိုရင် Tutorials လေးတွေသေချာလေ့လာလိုက်ရင်

အဆင်ပြေပါတယ်။

ကျွန်တော့ ဒီစာအုပ်နဲ့မလုံလောက်သေးဘူးထင်ရင်လည်း ဆက်လေ့လာလို့ရပါတယ်။

ကျွန်တော်ကတော့ အတတ်နိုင်ဆုံးတော့ အပြည့်အစုံဖြစ်အောင်ရေးထားပါတယ်။

ဒါဆိုရင်ဘယ်လိုရေးမလဲ။

ဘယ်လိုရေးမလဲဆိုတာကို မလေ့လာခင် Good Code (Coding Standard) တွေရဲ့သဘောသဘာဝကို

အောက်ဖော်ပြပါအတိုင်းခွဲထုတ်ကြည့်လိုက်ရအောင်။

၁.ယုံကြည်စိတ်ချရခြင်း

၂.ပြုပြင်ထိန်းသိမ်းဖို့လွယ်ကူခြင်း

၃.ကျွမ်းကျင်ခြင်း

ဆိုသကဲ့သို့ ယေဘုယျသုံးမျိုးခွဲလိုက်ပါတယ်။

Programmer တော်တော်များများကတော့သူတို့ရဲ့ App တွေကို Performance ကောင်းဖို့လောက်သာအလေးပေးရေးသားနေကြပါတယ်။

အဓိက ကသူတို့ရေးနေတဲ့ App ကိုလာအပ်ထားတဲ့သူရဲ့ ပြောတဲ့ပုံစံအတိုင်းသာလုပ်ဆောင်နေကြပါတယ်။

Standard Coding ရေးနည်းကိုမေ့ထားပြီးတော့ ရေးနေကြပါတယ်။

ပြောလိုက်ရင်လည်း ....ဘာဖြစ်လဲကွာ…အဓိက App အလုပ်လုပ်ရင်ပြီးတာဘဲမဟုတ်လားဆိုပြီးတော့ပြန်ပြန်

ဖြေနေကြတာကိုတွေ့ရပါတယ်။

ဒါကြောင့် သူတို့ရေးတဲ့အခါမှာ

၁.Performance ကောင်းစေရန်အတွက်

၂.အလုပ်အပ်တဲ့သူရဲ့ သဘောတူညီချက်အတွက်

၃.ယုံကြည်စိတ်ချဖို့အတွက်

၄.ပြုပြင်ထိန်းသိမ်းဖို့အတွက်

ဆိုပြီးတော့ အဆင့်လေးဆင့်အတိုင်းရေးနေကြပါတယ်။

ROI (Return On Investment) အရ ကတော့ နံပါတ်က ၃,၄,၁,၂ ဖြစ်ပါတယ်။

၁.ယုံကြည်စိတ်ချဖို့အတွက်

၂.ပြုပြင်ထိန်းသိမ်းဖို့အတွက်

၃.Performance ကောင်းစေရန်အတွက်

၄.အလုပ်အပ်တဲ့သူရဲ့ သဘောတူညီချက်အတွက်

ဆိုပြီးတော့ဖြစ်သင့်ပါတယ်။တကယ်လို့ Programmer တွေရဲ့ Coding ရေးထားတဲ့ပုံစံက

အဲ့ဒီလိုမဟုတ်ဘူးဆိုရင်တော့ နောင်အချိန်မှာ ပြန်လည်ပြုပြင်တာတို့...Maintain လုပ်တာတို့...

စတာတွေဖြစ်လာတဲ့အခါမှာ အချိန်များစွာကိုကုန်စေပြီးတော့ အကျိုးမရှိဖြစ်မှာသေချာပါတယ်။

တစ်နေရာရာကို ပြုပြင်ချင်လို့ဘဲဖြစ်စေ...Update လုပ်ချင်လို့ဘဲဖြစ်စေ...Coding Standard မဟုတ်တဲ့အတွက်ကြောင့် ကိုယ်ပြင်ချင်တဲ့နေရာကို လိုက်ရှာနေရတဲ့အတွက်ကြောင့် အချိန်များစွာကိုကုန်စေသလို အမှားများကိုလည်းတွေ့ကြုံစေမှာဖြစ်ပါတယ်။

ဒါကြောင့် Standard မဖြစ်ဘဲနဲ့ Coding ရေးခြင်းကသင့်ရဲ့ ဘဝအတွက်မကောင်းတဲ့အချက်ဖြစ်တဲ့အတွက်

ကြောင့် ကိုယ်ရေးတဲ့ Code တွေက Standard မဖြစ်ဘူးဆိုရင်တော့ ပြုပြင်ပြောင်းလဲသင့်ပါတယ်လို့ကျွန်တော်အကြံပြုပါတယ်။

ဒါဆိုရင်ကျွန်တော်တို့ ရေးတဲ့ Coding တွေအတွက် ဘယ်လို Standard ဖြစ်အောင်လုပ်ကြမလဲဆိုတာကို စပြီးတော့လေ့လာကြည့်လိုက်ရအောင်။

Coding Standard & Best Practices ? ?? ???? ???? ?

စနစ်ကျပြီးယုံကြည်အားထားရတဲ့ Application တွေကို Develop လုပ်ချင်တယ်ဆိုရင်တော့ Coding Standards နဲ့ Best Practices တွေကိုလိုက်နာရမှာဖြစ်ပါတယ်။

ဒီစာအုပ်ထဲမှာဖော်ပြထားတဲ့ Coding Standard နဲ့ Best Practices တွေကတော့ ကျွန်တော့်ရဲ့ အတွေ့အကြုံတစ်ခုထဲနဲ့ရေးထားတာမဟုတ်ဘဲနဲ့···. Microsoft ကချမှတ်ထားတဲ့ Guideline တွေ ...တခြား Developer

တွေကသူတို့ရဲ့အတွေ့အကြုံတွေ...စတာတွေကိုပေါင်းစပ်ရေးသားထားခြင်းဖြစ်ပါတယ်။

Programming လောကမှာ ချမှတ်ထားတဲ့ စည်းကမ်းတွေကတော့ အများကြီးပါဘဲ···သူတို့ချမှတ်ထားတဲ့စည်းကမ်းတွေကိုလိုက်နာချင်လည်းရပါတယ်...

မလိုက်နာချင်လည်းရပါတယ်..ဒါကတော့ကိုယ့်သဘောပါဘဲခင်ဗျာ···တကယ်လို့လိုက်နာချင်တယ်ဆိုရင်တော့ကိုယ်ကိုယ်တိုင်စာတွေဖတ်..ပြီးရင်တော့ အများဆုံးလိုက်နာထားတဲ့ စည်းကမ်းတွေကိုလိုက်နာမယ်ဆိုရင်တော့အကောင်းဆုံးဖြစ်ပါတယ်···။

စည်းကမ်း A ကိုလူဆယ်ယောက်လိုက်နာတယ်။

စည်းကမ်း B ကိုလူ ၅ ယောက်လိုက်နာတယ်ဆိုရင်တော့ ကျွန်တော်အကြံပေးချင်တာကတော့ လူအများစု လိုက်နာတဲ့ A ကိုလိုက်နာရင်အကောင်းဆုံးဖြစ်မယ်လို့အကြံပေးချင်ပါတယ်။

ဒါဆိုရင် ကျွန်တော်တို့ လိုက်နာရမဲ့ အချက်လေးတွေကို လေ့လာကြည့်လိုက်ရအောင်ဗျာ။

Naming Conventions & Standards

ကျွန်တော်တို့အခုလေ့လာမဲ့အပိုင်းကို နှစ်ပိုင်းခွဲပြီးတော့လေ့လာလို့ရပါတယ်။

အဲ့ဒါတွေကတော့

1.Pascal Casing

2.Camel Casing

ဆိုပြီးတော့လေ့လာရမှာဖြစ်ပါတယ်။ဒါဆိုရင်အဲ့ဒီအပိုင်းနှစ်ပိုင်းတွေကဘာတွေလဲ။

ဘာတွေလဲဆိုတော့

Pascal Casing ကိုလိုက်နာပြီးတော့ရေးမယ်ဆိုရင်

စာလုံးတွေရဲ့ ပထမဦးဆုံး စာလုံးကို Upper Case နဲ့ရေးပြီးတော့ ကျန်တာအားလုံးကို Lower Case နဲ့ရေးရမှာဖြစ်ပါတယ်။

Camel Casing ကိုလိုက်နာပြီးတော့ရေးမယ်ဆိုရင်

စာလုံးတွေရဲ့ပထမဦးဆုံး စာလုံးကို Lower Case နဲ့ရေးပြီးတော့ ကျန်တဲ့စာလုံးတွေရဲ့ပထမစာလုံးကို Upper Case နဲ့ရေးရမှာဖြစ်ပါတယ်။

ပိုပြီးတော့နားလည်အောင်ပြောရမယ်ဆိုရင် ဥပမာလေးနဲ့လေ့လာကြည့်လိုက်ရအောင်။

Pascal Casing

Eg. BackColor , ForeColor , GridView , TextBox , SqlDataSource

စသဖြင့်ရေးရမှာဖြစ်ပါတယ်။

Camel Casing

Eg. backColor , foreColor , gridView , textBox , sqlDataSource

စသဖြင့်ရေးရမှာဖြစ်ပါတယ်။

ဒါဆိုရင် အခုဒီနှစ်ခုကို ဘယ်နေရာမှာဘယ်ဟာကိုသုံးမလဲ...။

ဘယ်နေရာမှာဘယ်ဟာကိုသုံးမလဲဆိုရင် Class Name တွေ Method Name တွေကိုရေးတဲ့အခါမှာဆိုရင် Pascal Casing ကိုသုံးသင့်ပါတယ်။

Variable , Method Parameter တွေကိုရေးတဲ့အခါမှာတော့ Camel Casing ကိုသုံးပြီးတော့ရေးသင့်ပါတယ်။

ပိုပြီးတော့ရှင်းသွားအောင် ဥပမာလေးနဲ့ကြည့်လိုက်ရအောင်။

```
Public class PascalClass

{

    //

{
```

အဲ့ဒါကတော့ Pascal Casing ကိုသုံးပြီးတော့ရေးထားတဲ့ Class Name ဘဲဖြစ်ပါတယ်။

နောက်တစ်ခုကတော့ Pascal Casing ကိုသုံးပြီးတော့ ရေးထားတဲ့ Method Name ဘဲဖြစ်ပါတယ်။

```
Private void PascalMethod()

{

    //

}
```

နောက်တစ်ခုကတော့ Camel Casing ကိုသုံးပြီးတော့ Variable ကြေညာတာလေးကိုတစ်ချက်လောက်လေ့လာကြည့်လိုက်ရအောင်။

```
Private void PascalMethod(string name)
{
        string  name;
        int  salary;
        string address;                 // ဒါကတော့ Camel Casing ကိုသုံးပြီးတော့ကွည်ထားတာပါ။
}
```

ဆိုပြီးတော့ကြည်သင့်ပါတယ်။အဲ့ဒီလိုကြည်မှအမှန်ဖြစ်ပါတယ်။အောက်ကအတိုင်းဆိုရင်တော့ Coding Standard မဖြစ်ပါဘူး။

```
Private void PascalMethod()
{
        String nam;
        String add;
        Int sal;
}
```

တစ်ချို့ကတော့ အခုပြထားတဲ့အတိုင်းကြေညာကြပါတယ်။အဲ့ဒီလိုကြေညာရင်မှားပါတယ်။တစ်ချို့ကတော့ အများကြီးရေးရမှာပျင်းလို့တယ်။မပျင်းပါနဲ့လို့အကြံပေးချင်ပါတယ်။ဒါမှသာ မိမိရေးသားတဲ့ Code တွေက စံမှီနေမှာဖြစ်ပါတယ်။

ဒါကြောင့် Variables တွေကိုကြေညာမယ်ဆိုရင်တော့ အဓိပ္ပာယ်ပြည့်ဝတဲ့ စာလုံးများကိုသုံးပြီးတော့ကြေညာရမှာဖြစ်ပါတယ်...။

အတိုကောက်စာလုံးသုံးခြင်းကိုရှောင်ကြဉ်သင့်ပါတယ်။

.Net Interface အတွက်ကိုလည်း Camel Casing ကိုသုံးပြီးတော့ ကြေညာပါ..။(eg. IEntity)

အရင်အစောပိုင်းကာလ တွေတုန်းက Programmer တွေကသူတို့ရဲ့ Member Variables တွေအတွက် အရှေ့ဆုံးစာလုံးမှာဒီလိုလေးကြေညာခဲ့ကြပါတယ်။

```
string m_sName;

int nAge;
```

အဲ့ဒီလိုကြေညာခြင်းက တခြား Programming Languages တွေမှာတော့ မှန်ချင်မှန်ပါလိမ့်မယ်…။

.Net Programming Language တွေအတွက်တော့ မမှန်ဘူးလို့ပြောလို့ရပါတယ်။

Member Variables တွေကိုကိုယ်စားပြုလို့ m_ ကိုသုံးတာသည် .Net Coding Standard နဲ့မညီတဲ့အတွက်ကြောင့် ထိုသို့ကြေညာခြင်းကိုရှောင်သင့်ပါတယ်။

ဒါကြောင့် C#.Net မှာ Variables တွေကိုကြေညာတော့မယ်ဆိုရင်တော့ Camel Casing ကိုလိုက်နာပြီးတော့ကြေညာခြင်းကသာလျှင် Coding Standard ကိုလိုက်နာခြင်းဖြစ်ပါတယ်။

Local variables တွေကိုကြေညာရင်လည်း Underscore ထည့်ပြီးတော့မကြေညာသင့်ပါဘူး။

နောက်တစ်ချက်တော့ Variable ကြေညာတဲ့အခါမှာ စာလုံး ၁ လုံးထဲမကြေညာသင့်ပါဘူး။ဘယ်လိုလဲဆိုတော့

int i;

int y;

int z;

စသဖြင့်…မကြေညာသင့်ပါဘူး…။ဒါပေမဲ့ ခြွင်းချက်တစ်ခုတော့ရှိပါတယ်။

အဲ့ဒါကတော့ .Net ရဲ့ for looping ပါဘဲ…။

For looping ရေးတဲ့အခါမှာ ကျွန်တော်တို့တွေ int i ဆိုပြီးတော့ကြေညာလေ့ရှိပါတယ်။ဒီလိုလေးပါ။

For (int i , i <= 10 ; I ++)

ဆိုပြီးတော့ကြေညာလေ့ရှိပါတယ်။အဲ့ဒါမှန်ပါတယ်။

စာလုံးတစ်လုံးထဲကြေညာတာကို Camel Casing အရ for looping ထဲမှာဘဲကြေညာရပါမယ်။

Variables ကြေညာတဲ့အခါမျိုးမှာလည်း ဆင်တူလေးတွေမကြေညာသင့်ပါဘူး။

နောက်တစ်ချက်ကဘာလဲဆိုတော့ Local Variable ကို .Net မှာပါတဲ့ Keyword တွေနဲ့ဆင်တူ

မကြေညာသင့်ပါဘူး။

Boolean variables , Properties , method တွေကိုကြေညာမယ်ဆိုရင် is keyword လေးခံပြီးတော့

အောက်ကအတိုင်းကြေညာသင့်ပါတယ်။

Private boolean _IsFinished

နောက်တစ်ခုကတော့ File Name ပေးတဲ့အခါမှာဘယ်လိုပေးမလဲဆိုတာကို လေ့လာကြမှာဖြစ်ပါတယ်။

Give File Name with Standard

Coding Standard အရဖိုင်နာမည်ပေးတဲ့အခါမှာလည်း မိမိရေးသားမဲ့ Class Name နဲ့တူညီအောင်ပေးရမှာဖြစ်ပါတယ်။

ဥပမာအနေနဲ့ပြောရမယ်ဆိုရင်တော့ Class Name က HelloWorld ဆိုရင် File Name က HelloWorld ဆိုပြီးတော့ပေးရမှာဖြစ်ပါတယ်။

သူ့ကိုနာမည်ပေးတော့မယ်ဆိုရင်တော့ Pascal Casing ကိုသုံးပြီးတော့ပေးရမှာဖြစ်ပါတယ်။

Tab and Spacing with Standard

အရင်တုန်းကဆိုရင် Text Indent တွေခြားချင်တယ်ဆိုရင်တော့ တစ်ခါတလေ Space တွေပုတ်ပြီးတော့ခြားခဲ့ကြပါတယ်။

အဲ့ဒီလိုလုပ်တာက Coding Standard မဟုတ်ပါဘူး။

Text Indent လုပ်ချင်တယ်ဆိုရင်တော့ တစ်ခုသောနည်းလမ်းက Tab လုပ်ခြင်းသာဖြစ်ပါတယ်။

Space ကိုလုံးဝမသုံးသင့်ပါဘူး…။Visual Studio က Tab တစ်ချက်ကို Space 4 နဲ့ သတ်မှတ်ထားပါတယ်။

ဒါကလည်းသိပ်တော့ စိတ်ပူစရာမလိုပါဘူး…ဘာဖြစ်လို့လဲဆိုတော့ ကျွန်တော်တို့သုံးနေတဲ့ Visual Studio IDE ကအလိုအလျောက် ပြင်ပေးတဲ့အတွက်ကြောင့် အဆင်ပြေပါတယ်။

ကျွန်တော်ပြောချင်တဲ့သဘောက ဘေးမျဉ်းကနေ ခြားမယ်ဆိုရင် Tab နဲ့ ဘဲခြားသင့်တယ်

လို့ ဆိုလိုတာဖြစ်ပါတယ်။Space 4 ချက်ပုတ်ပြီး ခြားခြင်းကို ရှောင်ရမှာဖြစ်ပါတယ်။

Comment with Standard

အခုလေ့လာမဲ့ အကြောင်းကတော့ Comment လုပ်တဲ့အခါမှာလိုက်နာရမဲ့စည်းကမ်းတွေဖြစ်ပါတယ်။

အရင်တုန်းကလည်း Comment တွေပိတ်ချင်တယ်ဆိုရင် လည်းကြိုရင်ကြိုသလိုပိတ်ခဲ့ကြပါတယ်။

အခုတော့ စနစ်တကျပိတ်တာကိုလေ့လာကြမှာဖြစ်ပါတယ်။ဒါဆိုရင်ဘယ်လိုပိတ်မလဲ။

အရင်တုန်းက ဒီလိုပိတ်ခဲ့ဘူးနိုင်ပါတယ်။

```
//Declared Variables
    string name;
    int age;
    int phone;
```

အဲ့ဒီမှာတစ်ခုသတိထားမိသလားတော့မသိဘူး…။ဘာလဲဆိုတော့ Declared Variables ဆိုတဲ့ Comment ပိတ်ထားတဲ့စာသားလေးက အောက်ကကြေညာထားတဲ့ Variables တွေရဲ့ အရှေ့ကိုရောက်နေတာပါ။

အဲ့ဒီလိုကြေညာထားရင်တော့ Coding Standard ကိုမလိုက်နာထားဘူးလို့ဆိုရမှာဖြစ်ပါတယ်။

ဒါဆိုရင်ဘယ်လိုရေးမလဲ။ဘယ်လိုရေးမလဲဆိုတော့

ကိုယ်ပိတ်မဲ့ Comment က ကိုယ်ရေးထားတဲ့ Code နဲ့ Same Level ဖြစ်ရမှာဖြစ်ပါတယ်။ဘယ်လိုလဲဆိုတော့ အပေါ်ကအတိုင်း ဥပမာပေးမယ်ဆိုရင်

```
    //Declared Variables
    string name;
    int age;
    int phone;
```

အခဲ့ဒီလိုဖြစ်ရမှာဖြစ်ပါတယ်။အခဲ့ဒီလိုရေးမှမှန်မှာဖြစ်ပါတယ်။ဒါဆိုရင်ပိုပြီးတော့ရှင်းသွားအောင် Code များများနဲ့ လေ့လာကြည့်လိုက်ရအောင်…။

```
//Comment Testing
 String name = "Hello";
 DateTime currentTime = DateTime.Now;

 string fullMessage = "Hello " + name + " This Time is "+ currentTime.ToString();
 Console.WriteLine(fullMessage);
```

အခဲ့ဒီလိုရေးရင်တော့ သတ်မှတ်ထားတဲ့ စံကိုလိုက်နာတာဖြစ်ပါတယ်။ဒါကြောင့် အကျဉ်းချုပ်ပြောရမယ်ဆိုရင်တော့ Comment ပိတ်ချင်တယ်ဆိုရင်တော့ Code နဲ့ Same Level ဖြစ်ရင်ရပါတယ်။

နောက်တစ်ခုလေ့လာကြမှာကတော့ တွန့်ကွင်းလေးတွေကို Coding Standard နဲ့ကိုက်ညီအောင် ဘယ်လိုရေးမလဲဆိုတာကိုလေ့လာကြမှာဖြစ်ပါတယ်။

Curly braces with Standard

ကျွန်တော်တို့အခုလေ့လာကြမှာကတော့ Curly braces လို့ခေါ်တဲ့ တွန့်ကွင်းလေးတွေကို စံမှီအောင် ဘယ်လိုရေးသားအသုံးပြုကြမလဲဆိုတာဘဲဖြစ်ပါတယ်။

ဒီအကြောင်းကတော့ သိပ်အများကြီးပြောစရာမလိုဘူးထင်ပါတယ်။

တော်တော်များများ ရေးသားတဲ့ပုံစံက စံမှီတဲ့ပုံစံဖြစ်နေလို့ပါ။

ဘယ်လိုလဲဆိုတော့ ကျွန်တော်တို့ Class တွေ Method တွေရေးသားခဲ့တဲ့အခါမှာဒီလိုလေးရေးသားခဲ့ပါတယ်။

```
Private void PascalMethod(string name)
{
                // Code
}
```

အခဲ့ဒီလိုရေးသားရင်မှန်ပါတယ်။ကျွန်တော်တို့အသုံးပြုနေတဲ့ Visual Studio IDE ကလည်း Auto ချိန်ညှိပေးပါတယ်။

အခဲ့ဒီလိုရေးသားရင်တော့မှန်ပါတယ်။အောက်ကပုံစံအတိုင်းရေးသားရင်တော့ မှားပါတယ်။

```
Private void PascalMethod(string name){

        // Code

}
```

အခဲ့ဒီလိုရေးသားရင်တော့မှားပါတယ်။နောက်တစ်ခုပြောချင်တာကတော့ တွန့်ကွင်းလေးတွေကို Line တွေနဲ့ တစ်ကြောင်းထဲမရေးရပါဘူး။ဘယ်လိုလည်းဆိုတော့

```
if(....){// Code        }
```

အခဲ့ဒီလိုရေးရင်မှားပါတယ်။ဒါကြောင့် တွန့်ကွင်းလေးတွေကိုရေးမယ်ဆိုရင် စာကြောင်းနောက်တစ်ကြောင်းဆင်းပြီးတော့ရေးရင်အကောင်းဆုံးဖြစ်ပါတယ်။ဒီလိုလေးရေးပါ။

```
if(...)
{
        // Code
}
```

အခုဒီလိုလေးရေးရင်မှန်ပါတယ်။ if နဲ့ for အတွက်ကိုလည်း Curly braces ကိုတစ်ကြောင်းထဲမထားသင့်ဘဲနဲ့ တစ်ကြောင်းဆင်းပြီးတော့မှထားသင့်ပါတယ်။

Spacing using Standard

နောက်တစ်ခုကတော့ Operator လေးတွေကိုကြေညာပြီးသွားရင် Space ခြားတာလေးပါ။

Operator တွေကို မရေးခင်နဲ့….ရေးပြီးအချိန်တွေမှာ Space တစ်ချက် ခြားရမှာဖြစ်ပါတယ်။

ဒါကလည်း IDE က Support ပေးထားတဲ့အတွက်ကြောင့်အဆင်ပြေပါတယ်။

ဘယ်လိုလည်းဆိုတော့ ဥပမာအနေနဲ့ပြောရရင် ကျွန်တော်တို့ if condition စစ်တဲ့အခါမှာဒီလိုလေးရေးကြပါတယ်။

```
if(Result==0)
{
        for(int i=1;i<=10;i++)
        {
                // Code
```

```
        }
}
```

ဆိုပြီးတော့ရေးကြပါတယ်။အခုဒီလိုရေးသားထားရင်တော့မှားပါတယ်။ဘာဖြစ်လို့လဲဆိုတော့ Operator တွေရေးပြီးရင် Single Space ခြားမှသာလျှင် Coding Standard ဖြစ်မှာဖြစ်ပါတယ်။

```
if ( Result == 0 )
{
        for ( int i = 1 ; i <= 10 ; I++ )
        {
                //Code
        }
}
```

အခုဒီလိုရေးသားရင်တော့မှန်ပါတယ်။နောက်ထပ်လေ့လာမှာကတော့ Group ဖွဲ့ထားတဲ့ Code တွေကို ရှင်းလင်းသွားစေရန် Blank Line ပြုလုပ်ချက်ကိုလေ့လာကြမှာဖြစ်ပါတယ်။

Blank Line Separate using Standard

မြန်မာလိုပြောရရင်တော့ Code လေးတွေကို Group လေးတွေဖွဲ့ပြီးတော့ သတ်သတ်စီဖြစ်အောင် တစ်လိုင်းဆင်းတဲ့အကြောင်းအရာဖြစ်ပါတယ်။ဥပမာလေးနဲ့ကြည့်လိုက်ရအောင်။

```
String name = "Hello";
DateTime currentTime = DateTime.Now;

string fullMessage = "Hello " + name + " This time is "+ currentTime.ToString();

Console.WriteLine(fullMessage);
Console.ReadLine();
```

ပထမ Group ကတော့ Variables ကြေညာတဲ့ Group ဖြစ်တဲ့အတွက်ကြောင့် သပ်သပ်တစ်ခုထားပါတယ်။ ပြီးတော့ တစ်လိုင်းဆင်းပြီးတော့မှ fullMessage ကိုတစ်ကြောင်းထားတဲ့အတွက်ကြောင့် Group တစ်ခုလို့ မှတ်ယူရမှာဖြစ်ပါတယ်။

တစ်လိုင်းထပ်ဆင်းတဲ့အတွက်ကြောင့် Console Class Group ဖြစ်ပါတယ်။

Console တွေဘဲရေးထားတဲ့အတွက်ကြောင့် Console Group ကိုခေါ်နိုင်ပါတယ်။

အဲ့ဒီလိုမျိုး Coding တွေကို သတ်သတ်မှတ်မှတ် Group ခွဲထားတဲ့အတွက်ကြောင့် အမှားဖြစ်တဲ့အခါမှာလည်း ပြင်ဆင်ရင် မြန်မြန်ဆန်ဆန် အမှားရှာတွေ့မှာဖြစ်ပါတယ်။

ဒါဆိုရင် အမှားဆိုရင်ကောဘယ်လိုလဲ။အမှားရေးထားတဲ့ Code Group ကတော့

```
String name = "Hello";
DateTime currentTime = DateTime.Now;
string fullMessage = "Hello " + name + " This Time is "+ currentTime.ToString();
Console.WriteLine(fullMessage);
Console.ReadLine();
```

အဲ့ဒီလိုမျိုးဘဲဖြစ်ပါတယ်။အားလုံးကို တစ်ကြောင်းစီဘဲခြားရမှာဖြစ်ပါတယ်။

"ငါက ပိုရှင်းအောင်လို့ နှစ်ကြောင်း ခြားတယ်ကွ…. " ဆိုတာမျိုးဆိုရင်တော့ Coding Standard ကိုလိုက်နာခြင်းမဟုတ်တဲ့အတွက်ကြောင့်ရှောင်ကြဉ်သင့်ပါတယ်။Single Blank Line သာခြားရမှာဖြစ်ပါတယ်။

ဒါဆိုရင်ကျွန်တော်တို့နောက်တစ်ခုလေ့လာကြမှာကတော့ Code တွေကို တစ်စုတစ်စည်း ထဲနဲ့ Group ဖွဲ့တဲ့ အခါမှာသုံးတဲ့ Code လေးတွေအကြောင်းဘဲဖြစ်ပါတယ်။

#region & #endregion using Standard

Code တွေကိုအဆက်အစပ်ရှိတဲ့ Group ဖွဲ့ထားချင်ရင်တော့ #region နဲ့ #endregion ကိုသုံးပြီးတော့မှ ဖွဲ့

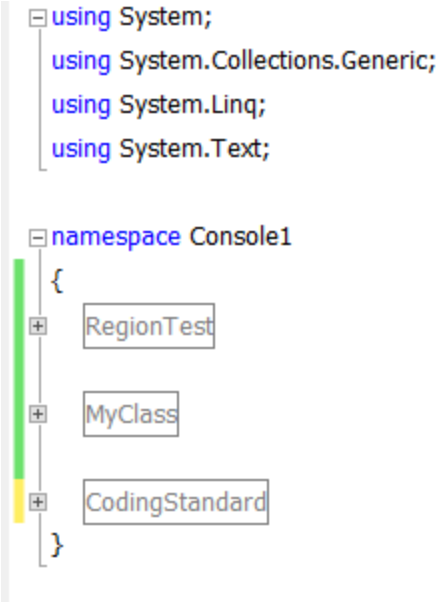
သင့်ပါတယ်။အဲ့ဒီလိုဖွဲ့ခြင်းက မိမိစိတ်ကြိုက်နာမည်ပေးလို့ရသလို စံမှီတဲ့ ရေးသားခြင်းလည်းဖြစ်ပါတယ်။ဒီလို လေးဖွဲ့လို့ရပါတယ်။

```
namespace Console1
{
    #region RegionTest
     class Region
    {
        public static void Main()
        {
```

```
            Console.WriteLine("Hello Buddy ");
        }
    }
    #endregion
}
```

အခုဒီလိုရေးသားခြင်းကြောင့် မိမိ၏ Code တွေကတစ်စုတစည်းထဲနဲ့ ရှင်းရှင်းလင်းလင်းဖြစ်နေပြီး စံမှီတဲ့ရေးသားချက်ဖြစ်မှာဖြစ်ပါတယ်။အပေါ်က Code လေးကို လည်း .Net ရဲ့ Collapsed လုပ်တဲ့အခါမှာလည်း RegionTest ဆိုပြီးတော့ မိမိပေးထားတဲ့နာမည်အတိုင်းမြင်ရမှာဖြစ်ပါတယ်။

အောက်ကပုံလေးကိုကြည့်လိုက်ရင်ရှင်းသွားမှာပါ။



နောက်တစ်ခုကတော့ Field တွေ Method တွေ Property တွေကိုကြေညာတဲ့အခါမှာဘယ်လို Standard ဖြစ်အောင်ကြေညာရမလဲဆိုတာကိုလေ့လာကြမှာဖြစ်ပါတယ်။

Declared Access Modifiers with Standard

ဘယ်လိုကြေညာရမလဲဆိုတာကို အောက်ကအဆင့်အတိုင်းလုပ်ဆောင်ကြေညာသင့်ပါတယ်။

1.Private Member Variables

2.Private Properties

3.Private Methods

4.Constructors

5.Public Properties

6.Public Methods

အထက်ပါဖော်ပြထားတဲ့အတိုင်းသာ အစီအစဉ်တကျကြေညာသင့်ပါတယ်။

သူကဘာနဲ့သွားတူသလဲဆိုတော့ OOP ရဲ့ Rules တွေနဲ့သွားတူပါတယ်။

OOP ရဲ့ Rules အရ Private AM ကိုအသားပေးထားပါတယ်။ပြီးမှသာ Public AM ကိုရေးပါတယ်။

ထို့အတူဘဲ Coding Standard မှာလည်းအတူတူပင်ဖြစ်ပါတယ်။ဒါဆိုရင်ကျွန်တော်တို့နောက်ထပ်လေ့လာမဲ့

အပိုင်းကတော့ UI (User Interface) တွေကိုနာမည် ပေးတဲ့အခါမှာ Standard ဖြစ်အောင်ဘယ်လိုပေးကြ

မလဲဆိုတာဘဲဖြစ်ပါတယ်။အရင်တုန်းက ကြိုရင်ကြိုသလိုပေးခဲ့အမှားမျိုးတွေရှိခဲ့ရင် အခုပြင်ဆင်ရမဲ့အချိန်

ရောက်လာပြီဖြစ်တဲ့အတွက်ကြောင့်ကျွန်တော်တို့လေ့လာကြည့်လိုက်ရအောင်။

Give UI Name with Standard

UI Name တွေကိုဘယ်လိုပေးမလဲ…။ဘယ်လိုပေးမလဲဆိုတော့ ကိုယ်ပေးချင်တဲ့ UI Element တွေနဲ့ သင့်လျော်တဲ့ နာမည်လေးတွေကိုပေးရမှာဖြစ်ပါတယ်။ထိုကဲ့သို့ပေးခြင်းအားဖြင့် ဘာကောင်းကျိုးတွေရသလဲဆိုတော့ ကိုယ်ကြေညာထားတဲ့(သို့) နာမည်ပေးထားတဲ့ Variables အားလုံးကိုလွယ်လွယ်ကူကူ Identify လုပ်နိုင်မှာဖြစ်ပါတယ်။

နာမည်ပေးလို့ရတဲ့ နည်းလမ်းနှစ်ခုရှိပါတယ်။

၁. UI Element တွေအတွက် ပထမစာလုံးကို (ui_) နဲ့စသင့်ပါတယ်။အဲ့ဒီလိုပြုလုပ်ခြင်းကဘာကောင်းသလဲဆိုတော့ UI Elements တွေကိုစနစ်တကျ Group ဖွဲ့ထားနိုင်ပါတယ်။ပြီးတော့ အဲ့ဒီ Group ဖွဲ့ထားတဲ့ UI ကို Access လုပ်ခြင်းလွယ်ကူသွားစေပါတယ်။

၂. UI Element တွေအတွက်ကိုယ်သုံးမဲ့ UI Element တွေရဲ့ကြိုတင်သတ်မှတ်ထားတဲ့ Prefix တွေကိုသုံးပြီးတော့လည်းနာမည်ပေးနိုင်ပါတယ်။အောက်မှာဖော်ပြထားတဲ့ Table လေးထဲမှာသက်ဆိုင်ရာ Prefix လေးတွေကိုဖော်ပြပေးထားပါတယ်။

Control	Prefix
Label	lbl
TextBox	txt

DataGrid	dtg
Button	btn
ImageButton	imb
Hyperlink	hlk
DropDownList	ddl
ListBox	lst
DataList	dtl
Repeater	rep
Checkbox	chk
CheckBoxList	cbl
RadioButton	rdo
RadioButtonList	rbl
Image	img
Panel	pnl
PlaceHolder	phd
Table	tbl
Validators	val

အပေါ်မှာပြထားတဲ့အတိုင်း ID တို့ Name တို့ပေးရင်တော့ Coding Standard ကိုလိုက်နာထားတဲ့အတွက်ကြောင့် စံမှီတဲ့ Naming တစ်ခုဖြစ်လာမှာဖြစ်ပါတယ်။

ဒါဆိုရင်ကျွန်တော်တို့နောက်ထပ်လေ့လာမယ့်အပိုင်းကတော့ Program များကိုရေးသားရာမှာ ကျင့်သုံးသင့်တဲ့ အချက်အလက်များကိုဆက်လက်လေ့လာရမှာဖြစ်ပါတယ်။

Good Programming Practices

Good Programming Practices ဆိုတာကတော့ Program တစ်ပုဒ်ရေးသားတဲ့အခါမှာဘယ်လိုရေးသင့်တယ်…ဘယ်လိုနေရာထားသင့်တယ်…စသည်ဖြင့်အမျိုးမျိုး

လေ့လာကြမဲ့အပိုင်းဖြစ်ပါတယ်။

1. Method ထဲမှာ Code တွေအများကြီးရေးသားခြင်းကိုရှောင်ကြဉ်ပါ။

Method တစ်ခုထဲမှာ Coding Line အရေအတွက်က 1~25 အထိဘဲရှိသင့်ပါတယ်။လိုင်းအရေအတွက်ကို ဆိုလိုတာပါ။စာလုံးအရေအတွက်မဟုတ်ပါဘူး။

အကယ်၍ မဖြစ်မနေ Method ထဲမှာလိုင်းအရေအတွက် 25 ထက်ပိုတော့မယ်ဆိုရင်တော့ ပိုတော့မယ့် လိုင်းကို နောက်ထပ် Method တစ်ခုခွဲရေးလိုက်ပါ။ဥပမာအနေနဲ့ပြောရရင်တော့

```
Private static void Main()
{
        //Output အပါအဝင် 25 ကြောင်းရေးပြီး…သို့ပေမဲ့ တစ်ကြောင်းရေးရန်ကျန်နေ
        Output(name);
}
```

```
Private static void Output(string name)
{
        Console.WriteLine(name);
}
```

ထိုကဲ့သို့ရေးရမှာဖြစ်ပါတယ်။အပေါ်က Method မှာ ၂၅ ကြောင်းပြည့်သွားတဲ့အတွက်ကြောင့် Output ထုတ်ချင်တဲ့အတွက်ကြောင့် ထပ်မရေးတော့ဘဲ နောက်ထပ် Method တစ်ခုကိုရေးလိုက်ပြီး Output ထုတ်လိုက်ခြင်းဖြစ်ပါတယ်။အလုပ်ရှုပ်တယ်လို့တော့မမြင်စေချင်ပါဘူး။

2.Method Name ပေးတော့မယ်ဆိုရင် ကိုယ်ပေးမဲ့ Method ကဘာလုပ်သလဲဆိုတာကို အရင်ကြည့်ပြီးမှပေးပါ။

ဥပမာအနေနဲ့ပြောရရင် ကိုယ့်ရဲ့ Method ထဲမှာရေးထားတဲ့ အကြောင်းအရာက Phone Number ကို Save တဲ့အကြောင်းကိုရေးထားတယ်ဆိုရင် ကိုယ်ပေးမဲ့ Method Name က သူနဲ့ကိုက်ညီတဲ့ Name ဘဲဖြစ်သင့်ပါတယ်။ SavePhoneNumber ဆိုပြီးတော့ဘဲဖြစ်သင့်ပါတယ်။ပိုရှင်းသွားအောင် ဥပမာလေးနဲ့ကြည့်လိုက်ရအောင်။

```
Private void SavePhoneNumber(int number)
{
        // SavePhoneNumber Code
}
```

အဲ့ဒီလိုရေးရင်မှန်ပါတယ်။Method ထဲမှာရှိတဲ့ Code တွေက Phone No ကို Save လုပ်တဲ့ Code တွေဆိုတော့ သူ့ရဲ့ Method Name ကို SavePhoneNumber လို့ပေးလိုက်တဲ့အတွက်ကြောင့် Method Name ကိုကြည့်ရုံနဲ့ သူ့အထဲက အကြောင်းအရာတွေကို ထင်ထင်ရှားရှား သိနိုင်ပါတယ်။

ဒါကြောင့် Method Name ကို အဆင်ပြေသလိုမပေးဘဲ…စနစ်တကျပေးတတ်ဖို့လိုပါတယ်။

3. Method တစ်ခုထဲမှာ Coding Standard အရ အလုပ်တစ်ခုဘဲလုပ်သင့်ပါတယ်။ဘယ်လိုလဲဆိုတော့ ကိုယ် ရေးထားတဲ့ Method က Phone No Save တဲ့အလုပ်ကိုလုပ်တယ်ဆိုရင် Phone No Save တဲ့အလုပ်တစ်ခုထဲ ကို ဘဲရေးထားသင့်ပါတယ်။

အဲ့ဒီအထဲမှာ Email Send လုပ်တဲ့ Code တွေကိုမရေးထားသင့်ပါဘူး။

Email Send လုပ်တဲ့အလုပ် တောင်မဟုတ်ပါဘူး…။

အလုပ်သေးသေးလေးဆိုရင်တောင် မလုပ်သင့်ပါဘူး…။

ပိုရှင်းသွားအောင် ဥပမာလေးနဲ့ကြည့်လိုက်ရအောင်။

```
string phonenumber;

string email;


SavePhoneNo(phonenumber);

SendEmail(email);


Private void SavePhoneNo(string no)
```

```
{
        // Code that save PhoneNumber
}

Private void SendEmail(string mail)
{
        // Code that send Email
}
```

အခုဒီလိုရေးထားရင် စံ ကိုလိုက်နာထားပါတယ်။

ဘာဖြစ်လိုလဲဆိုတော့ Method တစ်ခုက အလုပ်တစ်ခုဘဲလုပ်တဲ့အတွက်ကြောင့် ဖြစ်ပါတယ်။

ဒါဆိုရင် အများလေးကိုဆက်ကြည့်လိုက်ရအောင်။

```
String phonenumber;
String email;

SaveNo(phonenumber,email);

Private void SaveNo(string no,string mail)
{
        // Code that save PhoneNumber and Send Email
```

```
        two functions in one method (that's wrong)
}
```

အလုပ်နှစ်ခုကို Method တစ်ခုထဲမှာပေါင်းလုပ်ထားတဲ့အတွက်ကြောင့် ထိုသို့ပြုလုပ်ခြင်းက Coding Standard ကိုမလိုက်နာတဲ့အတွက်ကြောင့် စံမှီတဲ့ရေးသားနည်းလို့မဆိုနိုင်ပါဘူး။

ဒါဆိုရင် ကျွန်တော်တို့ ဆက်လေ့လာကြမှာကတော့ Data Type တွေကိုကြေညာရင်
ဘယ်လိုစနစ်တကျကြေညာသင့်သလဲဆိုတာကို ဆက်လေ့လာကြမှာဖြစ်ပါတယ်။

ကျွန်တော်တို့ လေ့လာကြည့်လိုက်ရအောင်...။

Declared DataType using Standard

4.ဒါကလည်းအထွေအထူးပြောစရာတော့မလိုဘူးလို့ထင်ပါတယ်။ဘာဖြစ်လို့လဲဆိုတော့ Programmer အများစုကြေညာနေကြတဲ့ ပုံစံက များသောအားဖြင့်မှန်လို့ပါ။ကျွန်တော်တို့ Data Type တွေကိုကြေညာတဲ့အခါမှာ ဒီလိုလေးကြေညာကြပါတယ်။

```
int count;

int age;

string name;

object mytype;
```

ဆိုပြီးတော့ကြေညာခဲ့ကြပါတယ်။အဲ့ဒီလိုကြေညာခဲ့ရင်တော့ Coding Standard ကိုလိုက်နာပါတယ်။

Int32 , String , Object စသဖြင့်မကြေညာသင့်ပါဘူး။

5.နောက်တစ်ခုကတော့ မထင်မှတ်ထားတဲ့ တန်ဖိုးတွေကို အမြဲသတိထားသင့်ပါတယ်။ဥပမာအနေနဲ့ပြောရ

မယ်ဆိုရင်တော့ တန်ဖိုးနှစ်ခုလက်ခံတဲ့ Parameter နှစ်ခု ကိုသုံးနေပါတယ်။

အဲ့ဒီလိုသုံးနေတဲ့အခါမှာ ကိုယ်ရေးထားတဲ့ Code တွေထဲကို ဝင်လာတဲ့ တန်ဖိုးနှစ်ခုမှာ ကိုယ်ထင်ထားသလို

တန်ဖိုးဘဲဝင်နေမယ်လို့မျှော်လင့်နေပါတယ်။ ဘယ်လိုလဲဆိုတော့ ကိုယ်က ပထမ Parameter အတွက် ၁ နဲ့

ဒုတိယ Parameter အတွက် ၂ ဆိုပြီးတော့ မျှော်လင့်ထားပါတယ်။ပြောရရင် ကိုယ်ရေးထားတဲ့ Code တွေထဲ

ကို မည်သည့် ကိန်းဂဏန်းမဆိုဝင်လာလို့ရပါတယ်။

ကိုယ်ကအဲ့ဒီလိုထင်ပါတယ်။အဲ့ဒီလိုထင်နေတဲ့အချိန်မှာ အသုံးပြုသူ User ကမသိဘဲနဲ့ ကိန်းမထည့်ဘဲ စာသား

(Character) ထည့်လိုက်တဲ့အခါမှာ ကိုယ်ရေးထားတာက ကိန်းအတွက်ဆိုတော့ Error ထတက်ပါတယ်။ Error

တက်တာက အကြောင်းတော့မဟုတ်ပါဘူး။Error တက်ပြီးတော့ Program ပိတ်သွားတာမျိုးဖြစ်နိုင်ပါတယ်။

ဒါဆိုရင် Program ထဲမှာကိုယ်လုပ်ထားတဲ့ အရာတွေက ပျောက်သွားနိုင်တဲ့အတွက် အသုံးပြုသူ User က

အဆင်မပြေဖြစ်နိုင်ပါတယ်။

အထက်ပါဖော်ပြချက်တွေကြောင့် သတိထားသင့်ပါတယ်။Coding Standard အရ အထက်ဖော်ပြချက်အတိုင်း

မဖြစ်စေရန်အတွက် လိုက်နာရမဲ့ အချက်လေးတွေကိုဖော်ပြပေးထားပါတယ်။

ပိုပြီးရှင်းသွားအောင် ကျွန်တော်တို့ ဥပမာလေးနဲ့လေ့လာကြည့်လိုက်ရအောင်ဗျာ…။

အောက်က ဥပမာလေးကတော့ Coding Standard အရလုပ်ဆောင်သင့်တဲ့အချက်ဘဲဖြစ်ပါတယ်။

```
If ( MemberType  ==  eMemberTypes.Registered )
{
        // Code that do something for Registered Users
}

Else if ( MemberType  ==  eMemberTypes.Guest )
{
        // Code that do something for Guest Users
}

Else
{
        //  Error Throw and handling that Error
```

ဤသို့ပြုလုပ်ခြင်းကြောင့် ဘာကြောင့် Error တက်သလဲဆိုတာကိုအလွယ်တကူသိနိုင်သလို နောင်တစ်ချိန်မှာ User အသစ်တွေကို ပြုလုပ်ခဲ့အခါမှာတက်နိုင်တဲ့ Error တွေကိုလည်းအလွယ်တကူ ရှာဖွေတွေ့ရှိနိုင်ပါတယ်။

```
}
```

အခုဒီလိုရေးရင်လည်းရပါတယ်။ဒါမှမဟုတ် Error Handling လုပ်တဲ့ try Block တို့ catch Block နဲ့လုပ်ရင် Coding Standard ကိုလိုက်နာတာဖြစ်ပါတယ်။

ဒါဆိုရင် ဘယ်လိုရေးရင်မလိုက်နာဘူးလည်းဆိုတာကိုပါလေ့လာကြည့်လိုက်ရအောင်။

```
If ( MemberType == eMemberTypes.Registered )
{
        // Code that do something for Registered Users
}

Else
{
        // Code that do something for Guest users and Unkown users
        ဤသို့ပြုလုပ်ချင်းကြောင့် Error တက်ခဲ့အခါမှာ Handle ပြုလုပ်ရမလွယ်သလို နောင်တစ်ချိန်မှာ User အသစ်တွေကို ပြုလုပ်ခဲ့အခါမှာတက်နိုင်တဲ့ Error တွေကိုလည်းအလွယ်တကူ ရှာဖွေတွေ့ရှိနိုင်မှာမဟုတ်ပါဘူး။
}
```

အထက်ပါအတိုင်းရေးထားခြင်းကြောင့် Error တက်တဲ့အခါမျိုးမှာ အမှားရှာရင် လွယ်လွယ်ကူကူရှာတွေ့နိုင်မှာမဟုတ်ပါဘူး။

Don’t use Hardcode

6.Hardcode တွေကိုတတ်နိုင်ရင်ဘယ်တော့မှမသုံးပါနဲ့။သူ့ကိုသုံးမဲ့အစား Constants ကိုသုံးပါ။

Constants ကို File ရဲ့အပေါ်မှာကြေညာပြီးတော့မှ ကိုယ်သုံးမဲ့ Code ထဲမှာယူသုံးပါ။

ဒါပေမဲ့ Constants ကို File အပေါ်မှာကြေညာပြီးတော့သုံးခြင်းကလည်း Coding Standard အရ စံမှီတဲ့ ရေးနည်းတော့မဟုတ်ပါဘူး။

Pure Coding Standard ကိုသုံးချင်တယ်ဆိုရင်တော့ ကိုယ်ရေးမဲ့ Constants ကို File အထဲမှာမကြေညာဘဲနဲ့ Configuration File ထဲမှာဖြစ်ဖြစ် Database ထဲမှာဖြစ်ဖြစ်ကြေညာထားပြီးမှမိမိရေးမဲ့ Code ထဲမှ ယူသုံးခြင်းကတော့ အမှန်တကယ် Coding Standard ကိုလိုက်နာထားသော ရေးနည်းသာဖြစ်ပါတယ်။

အဲ့ဒီလိုပြုလုပ်ခြင်းကဘာကောင်းကျိုးရသလဲဆိုတော့ နောင်တစ်ချိန်မှာပြန်ပြင်ချင်တဲ့အချိန်မျိုးရောက်ရင် ဖိုင်တွေအားလုံးကို လိုက်ပြင်နေစရာမလိုတော့ဘဲ ဖိုင်တစ်ခုထဲက Code ကိုပြင်လိုက်တာနဲ့သူ့ကိုသုံးထားတဲ့ ဖိုင်တွေအားလုံးကိုပြင်ပြီးသားဖြစ်သွားပါတယ်။

7.နောက်တစ်ခုကတော့ HardCode String တွေကိုမသုံးဘဲနဲ့ Resource Files တွေကိုသုံးပါ။

Configuration file , XML , Database တို့စသဖြင့်ကိုသုံးပါ။

8. string တွေကို တစ်ခုနဲ့တစ်ခု မနှင်းယှဉ်မီ Lower Case (or) Upper Case ကိုပြောင်းပြီးမှ တစ်ခုနဲ့တစ်ခု Compare လုပ်ပါ။ဒါမှသာ string တွေကို မတူညီတဲ့ Case တွေနဲ့ Compare လုပ်တဲ့အခါမှာ ပိုပြီးတော့ အဆင်

ပြေမှာဖြစ်ပါတယ်။

```
If ( name.ToLower() == “hello” )
{
        // Code
}
```

9. တစ်ခါတစ်လေ ကျွန်တော်တို့ TextBox ထဲမှာ ရှိတဲ့စာသား (Character) တွေကိုရှင်းထုတ်ချင်တယ်ဆိုရင်

ကျွန်တော်တို့ txt.Text=” ”; ဆိုပြီးတော့ ဘာမှမရှိတဲ့ string ကိုထည့်ပြီးတော့ ရှင်းခဲ့ဖူးပါတယ်။

Coding Standard ကိုလိုက်နာမယ်ဆိုရင်တော့ အဲ့ဒီလိုမရေးသင့်ပါဘူး။

အဲ့ဒီလိုရေးမဲ့အစား .Net မှာအသင့်ပါတဲ့ string.Empty ကိုသုံးပါ။ပိုရှင်းသွားအောင် ဥပမာလေးနဲ့ကြည့်လိုက်ရအောင်။

```
If ( name == string.Empty )
{
        // Code that do something
}
```

အဲ့ဒီလိုရေးရင် စံ ကိုလိုက်နာထားပါတယ်။

```
If  ( name == " " )
{
        // Code that do something
}
```

အခဲ့ဒီလိုရေးရင်တော့ မလိုက်ဘူးလို့ဆိုနိုင်ပါတယ်။

10. Member Variables တွေသုံးခြင်းကို ရှောင်ပါ။ထို့ကြောင့် မိမိလိုအပ်တဲ့ နေရာရောက်မှဘဲ Local Variables တွေကိုကြေညာပြီးတော့ လိုသလိုသုံးပါ။ Member Variables ကိုကြေညာထားပြီး Method တွေ အများကြီးက ယူသုံးနေရင် မလိုလားအပ်တဲ့ Error တွေတက်နိုင်ပါတယ်။အကယ်၍ Member Variables ကိုကြေညာထားပြီး Method တွေက Share လုပ်ပြီးယူသုံးတဲ့အခါမှာ ဘယ် Method က ဘယ်အချိန်မှာ ဘယ် နေရာကို ပြောင်းလဲသွားသလဲဆိုတာကို Track လိုက်ကြည့်တဲ့အခါမှာ ခက်ခဲသွားမှာဖြစ်ပါတယ်။

11. enum ကို အလျဉ်းသင့်သလိုသုံးပါ။ သိသာထင်ရှားပြီး မတူညီတဲ့ တန်ဖိုးတွေကို ဖော်ပြချင်တယ်ဆိုရင် string (or) number တွေကိုမသုံးပါနဲ့။ဥပမာနဲ့ပြောရရင် Gender ဆိုရင် Male နဲ့ Female ဘဲရှိတဲ့အတွက် သူ့ကိုကြေညာရင် string နဲ့မကြေညာဘဲနဲ့ .Net မှာပါတဲ့ enum ကိုသုံးပြီးတော့ကြေညာသင့်ပါတယ်။

ပိုပြီးတော့ ရှင်းသွားအောင် ဥပမာလေးနဲ့ကြည့်လိုက်ရအောင်…။ဥပမာ မှာတော့ HTML လေးနဲ့ဘဲပြထားပါ တယ်။

```
enum Email
	{
		Html,
		PlainText,
		Attachment
	}

	void SendMail (string message, Email email)
	{
		switch ( email )
		{
			case Email.Html:
				// Code that Do something
				break;
			case Email.PlainText:
				// Code that Do something
				break;
			case Email.Attachment:
				// Code that Do something
				break;
			default:
				// Code that Do something
				break;
		}
	}
```

အခုဒီလိုရေးထားရင်တော့ Coding Standard ကိုလိုက်နာထားတဲ့အတွက်ကြောင့် စံမှီတဲ့ ရေးနည်းဘဲဖြစ်ပါတယ်။

ဒါဆိုရင် စံ ကိုမလိုက်နာတဲ့ ပုံစံကတော့

```
void SendMail (string message, string email)
        {
                switch ( email )
                {
                        case "Html":
                                // Do something
                                break;
                        case "PlainText":
                                // Do something
                                break;
                        case "Attachment":
                                // Do something
                                break;
                        default:
                                // Do something
                                break;
                }
}
```

12. Member Variables တွေကို public နဲ့ protected နဲ့မကြေညာသင့်ပါဘူး။ကြေညာမယ်ဆိုရင်တော့ OOP Rules ရဲ့ စံအရ Private နဲ့ဘဲကြေညာသင့်ပါတယ်...။ပြီးတော့မှ public Property ကိုသုံးပြီးတော့ပြောင်းလဲသင့်ပါတယ်။

13. Event Handler method ထဲမှာ Required Action ကိုဆောင်ရွက်တဲ့ Code တွေမပါသင့်ပါဘူး။

နောက်ပြီးတော့ ကိုယ်ရေးထားတဲ့ Event Handler Method ကနေနောက်ထပ် method တစ်ခုကိုခေါ်တာမျိုးတွေကိုပြုလုပ်သင့်ပါတယ်။ Method တစ်ခုထဲမှာဘဲ အားလုံးမရေးဘဲနဲ့သီးခြား Method တွေကို တစ်ခုချင်းခေါ်သုံးတာမျိုးတွေ ပြုလုပ်ခြင်းကိုဆိုလိုတာဖြစ်ပါတယ်။

14. မိမိရေးသားနေတဲ့ Code ထဲမှာပါတဲ့ Path လမ်းကြောင်းတွေကိုပေးတဲ့အခါမှာလည်း Hardcode တွေကိုသုံးပြီးတော့မပေးပါနဲ့။

နောက်ပြီးတော့ Drive Name တွေကိုလည်း HardCode မပေးပါနဲ့။

Application Path တွေကို Relative Path နဲ့ Programmatically ဘဲပေးပါ။

15. ကိုယ်ရေးထားတဲ့ Code တွေက Hardcode ဖြစ်နေရင် အခက်အခဲကြုံနိုင်ပါတယ်။ဘယ်လိုမျိုးလဲဆိုတော့ ကိုယ်က Drive D: ထဲမှာ Data တွေသွားသိမ်းအောင် Hardcode နဲ့ရေးထားပါတယ်။

ထို့ကဲ့သို့ပြုလုပ်ခြင်းမျိုးကိုရှောင်ပါ။ဘာဖြစ်လို့လဲဆိုတော့ ကိုယ့်စက်မှာကတော့ D: ထဲမှာသွားသိမ်းလို့ရပေမဲ့ တခြားသူများစက်မှာ D: မရှိဘဲနဲ့ A: , B: , Z: တို့စသည်ဖြင့်ရှိလာတဲ့အခါမှာ Error တက်လာနိုင်တဲ့အတွက်ကြောင့် ရှောင်ကြဉ်ရပါမယ်။

16. ကိုယ်ရေးထားတဲ့ Application ကိုစမ Run ခင်မှာ ကိုယ်ရေးထားတဲ့ Application ကို Run နိုင် မဲ့ Drive Location မှာ ကိုယ်ရေးထားတဲ့ Application အလုပ်လုပ်စေနိုင်ဖို့အတွက်

ဖိုင်တွေရှိမရှိကို ဦးစွာ စစ်တဲ့ Code တွေကိုရေးသားထားဖို့လိုပါတယ်။

ဥပမာအနေနဲ့ပြောရရင် Database နဲ့ချိတ်ဆက်

တဲ့ Code တွေပါခဲ့ရင် အရင်ဆုံးဦး ကျွန်တော်တို့သုံးမဲ့ Database Connetion ရှိမရှိ...အလုပ် လုပ်...

မလုပ် စသဖြင့်စစ်တဲ့ Code တွေကိုထည့်ထားဖို့လိုပါလိမ့်မယ်။

အကယ်၍ Problem (Error) တက်ခဲ့တယ်ဆိုရင်လည်း အသုံးပြုသူ User ကို Message Box လိုမျိုး

ရိုက်မပြဘဲနဲ့...ခင်မင်ရင်းနှီးတဲ့ စကားလုံးတွေနဲ့ Error တက်နေကြောင်းကိုပြသသင့်ပါတယ်။

ဘာဖြစ်လို့ Message Box ကမကောင်းရသလဲဆိုတော့ လေ့လာတွေ့ရှိချက်များအရ Message Box

ဖြင့်ပြ ခြင်းက User ကိုအထိတ်တလန့်ဖြစ်စေတယ်လို့ဆိုထားပါတယ်။ဥပမာ.. သုံးနေတုန်း (ဒုန်) ဆို

ပြီးတော့ တက်လာရင် User ကအထိတ်တလန့်ဖြစ်စေတဲ့အပြင် စိတ်ပျက်သွားနိုင်တဲ့အတွက်ကြောင့် Label လေးကို အရောင်လေးနဲ့ဖြစ်စေ... ဘေးမှာ အရောင်လေးနဲ့ဖြစ်စေ ယဉ်ကျေးစွာရေးသား အသိ ပေးသင့်ပါတယ်။

17. အကယ်၍ ကိုယ်ရေးထားတဲ့ Code တွေထဲမှာ Configuration File ပါတဲ့အခါမှာ

Configuration File လိုအပ်တဲ့အခါမျိုးမှာ မရှိခဲ့ဘူးဆိုရင် Default Value နဲ့ Configuration ဖိုင် အသစ်တစ်ခုဆောက်ပေးထားဖို့လိုပါတယ်။

18. အကယ်၍ Configuration File ထဲမှာ မှားနေတဲ့ တန်ဖိုးကို တွေ့ရှိလို့ဘဲဖြစ်စေ...ထည့်

လိုက်တဲ့

တန်ဖိုး မှားယွင်းနေလို့ဘဲဖြစ်စေ...Label သို့ Error Throw လုပ်ပြီးတော့ဘဲဖြစ်စေ... အသုံးပြုသူကိုဖော်ပြပေးဖို့လိုပါတယ်။ထပ်ပြီးတော့ အသုံးပြုသူ User ကို ထည့်ရမဲ့တန်ဖိုး ကို Sample

အနေနဲ့လည်းဖော်ပြပေးဖို့လိုပါလိမ့်မယ်။

19. Error Message ကအသုံးပြုသူကို Error ဖြေရှင်းနိုင်စေရန် များစွာအကူအညီပေးပါတယ်။

ထို့ကြောင့် User ကို ဖော်ပြတဲ့အခါမျိုးမှာ " Error in Application " , " There is an Error " ဆိုပြီးတော့ဘယ်တော့မဖော်ပြဖို့လိုပါတယ်။ထိုသို့ဖော်ပြလိုက်တာနဲ့ User ကဘယ်နေရာမှာ

ဘယ်လိုသွားပြင်ရ မလဲဆိုတာ မသိတော့တဲ့အခြေအနေမျိုးကြုံတွေ့နိုင်ပါတယ်။

ဥပမာအနေနဲ့ပြောရရင် Database ကို Update သွားလုပ်တဲ့ Code တွေမှာဆိုရင် Error တက်တဲ့အခါ

မှာ ပြသင့်တာကတော့ " Fail to Update Database " ဆိုပြီးတော့ပြသင့်ပါတယ်။

20.Login ဝင်တဲ့အခြေအနေမျိုးတွေမှာဆိုရင်လည်း မှားနေရင် " Please make sure your

username and Password" တို့.... " username and password are do not match "

စသဖြင့်ပြသင့်ပါတယ်။

ဤသို့ပြခြင်းကြောင့် User ကလည်း အော်...ငါမှားနေတာဒါပါလားဆိုတာကို သိပြီးတော့ မှ

အမှန်ကိုပြင်နိုင်မှာဖြစ်ပါတယ်။

21. နောက်တစ်ခုကတော့ Error ပြတဲ့အခါမှာ Programming ဆိုင်ရာ Code တွေ... Logic တွေနဲ့ မပြ ဖို့တော့လိုပါလိမ့်မယ်။ဘာကြောင့်လည်းဆိုတော့ Programming ဆိုင်ရာ Code တွေက

Programmer တွေနဲ့ရင်းနှီးကျွမ်းဝင်ထားတဲ့ သူတွေမှဘဲသိမှာဖြစ်ပါတယ်။

လူအထင်ကြီးအောင် ငါ့ Program က Programming Code နဲ့ဘဲ Error ပြလို့ အဆင့်မြင့်တယ်လို့ မထင်ပါနဲ့။ရှောင်ကြဉ်သင့်ပါတယ်။

Error အကြောင်းကို အားလုံးခြုံပြီးတော့ပြောရရင် အလွယ်ဆုံးနဲ့ အဆင်ပြေဆုံးဖြစ်အောင် Message ကိုပြတတ်ဖို့တော့လိုပါလိမ့်မယ်။

အဲ့ဒီလိုပြခြင်းက ဘာနဲ့တူသလဲဆိုတော့ စာရေးသူနဲ့တူပါတယ်။စာအုပ်တွေ...စာတမ်းတွေ...စာသင်တာတွေ...စတဲ့အလုပ်ကိုလုပ်တဲ့သူတွေဆိုရင်တော့ပိုသိနိုင်ပါတယ်။

စာရှင်းတယ်ဆိုတာ ကိုယ်နားလည်သလို ရှင်းလို့မရပါဘူး။ ကိုယ်ကိုယ်တိုင်က တကယ်တတ်ပေမဲ့ ကိုယ်နားလည်သလို သွားရှင်းရင်တော့ နားထောင်တဲ့သူတွေက ဘာမှနားလည်မှာမဟုတ်ပါဘူး။

ဒါကြောင့် စာရေးသားခြင်း...စာရှင်းခြင်းဆိုတာတွေကတော့ မိမိတတ်သမျှ သိသမျှ တွေကို အလွယ်ကူဆုံး အရိုးရှင်းဆုံး ဖြစ်အောင်..ရေး...ရှင်းပြ...စသည်ဖြင့်လုပ်သင့်ပါတယ်။

ရှင်းရှင်းပြောရရင် ကိုယ်က အရမ်းတော်တဲ့ ဆရာကြီးဘဲဖြစ်နေအုံးတော့..စာရှင်းရင်တော့ လူအထင်ကြီးအောင် စကားလုံးကြီးကြီးတွေနဲ့မရှင်းဘဲ.... လူပိန်း နားလည်အောင် စကားလုံးပေါ့ပေါ့ပါးပါးနဲ့ ရှင်းပြသင့်ပါတယ်။စာရေးနေတဲ့သူတွေကို ဆရာလုပ်တာမဟုတ်ပါဘူး။

ကျွန်တော်ကိုယ်တိုင်ကလည်း စာရေးဆရာ မဟုတ်တဲ့အတွက်ကြောင့် တတ်နိုင်သလောက်တော့ အားလုံးနားလည် စေရန် လွယ်လွယ်ကူကူ ရှင်းပြထားပါတယ်။

ထို့အတူဘဲ Error Message ပြတဲ့အခါမှာလည်း အလွယ်ကူဆုံးနဲ့ အရိုးရှင်းဆုံး ထိထိမိမိ ပြတတ်ဖို့တော့လိုပါတယ်။ဒါမှသာလျှင် Coding Standard ကိုလိုက်နာထားတဲ့အတွက်ကြောင့် စံ မှီတဲ့ရေးသား

နည်းဖြစ်ပါတယ်။

22. Single File တစ်ခုထဲမှာ တတ်နိုင်ရင် Class ကို တစ်ခုထပ်ပိုပြီးတော့မထားပါနဲ့။အဲ့ဒီလိုထားခြင်း က Coding Standard နဲ့မကိုက်ညီတဲ့အတွက်ကြောင့်ဖြစ်ပါတယ်။

23. File တစ်ခုထဲမှာ Coding တွေအများကြီးရေးခြင်းကိုရှောင်ကြဉ်သင့်ပါတယ်။ဖိုင်တစ်ဖိုင်ထဲမှာ လိုင်းရေ 1000 Lines ထက်မကျော်သင့်ပါဘူး..။ကျော်တော့မယ်ဆိုရင်လည်း နောက်ထပ် Class တွေ ထပ်ခွဲပြီးတော့ ရေးသင့်ပါတယ်။

24. public Method ရေးသားခြင်းကိုရှောင်ကြဉ်သင့်ပါတယ်။ဘာကြောင့်လဲဆိုတော့ public ရဲ့ သဘောအရ ကြိုက်တဲ့ Class ကအလွယ်တကူယူသုံးလို့ရပါတယ်။Same Assembly ထဲမှာဆိုရင်တော့ internal ကိုသုံးပြီးတော့ကြေညာရင်လည်း အဆင်ပြေပါတယ်။private ကိုသုံးရင်လည်းအဆင်ပြေပါတယ်။

25. Method တစ်ခုထဲကို Parameter တွေအများကြီး Pass လုပ်ခြင်းကိုလည်းရှောင်ကြဉ်ရပါမယ်။အကယ်၍ Parameter လေးခု..ငါးခု ရှိတော့မယ်ဆိုရင်တော့ Class တစ်ခု (သို့) Structure တစ်ခုကို Define လုပ်ဖို့အတွက်ပိုပြီးတော့သင့်တော်ပါတယ်။

26. အကယ်၍ ကိုယ်ရေးထားတဲ့ Method အထဲမှာ Collection Return ပြန်တဲ့အကြောင်းပါနေတယ်ဆိုရင် တစ်ခါတစ်လေ ပြန်စရာတန်ဖိုးမရှိတဲ့အခါမျိုးနဲ့ကြုံရနိုင်ပါတယ်။

အဲ့ဒါဆိုရင်ကျွန်တော်တို့အများ ဆုံးပြုလုပ်ကြတာကတော့ null တန်ဖိုးကို Return ပြန်တာဖြစ်ပါတယ်။

အမှန်တကယ် Coding Standard ကိုလိုက်နာမယ်ဆိုရင်တော့ အဲ့ဒီလို null return ပြန်တယ်ဆိုတာမှားပါတယ်။

ထို့ကြောင့် Coding Standard အရ ဘာမှ Return ပြန်စရာမရှိဘူးဆိုရင် null Return ပြန်မဲ့အစား

Empty Collection ကြီးကိုသာ ဒီအတိုင်း Return ပြန်ပေးလိုက်ပါ...။

ဥပမာအနေနဲ့ပြောရမယ်ဆိုရင် ArrayList Return ပြန်တဲ့ Method တစ်ခုရေးထားတယ်...။

ဒါပေမဲ့ Return ပြန်စရာကဘာမှမရှိဘူးဖြစ်နေတယ်။ဒါဆိုရင် ကျွန်တော်အပေါ်မှာပြောထားသလိုဘဲ

null Return မပြန်ပါနဲ့။ သုည (0) ကိုသာ Return ပြန်ပေးလိုက်ပါ။

အဲ့ဒီလို ပြုလုပ်ခြင်းကြောင့် ဘာတွေကောင်းကျိုးရသလဲဆိုတော့ Application Calling ပြုလုပ်တဲ့အခါမှာ Count ကို Check ပြုလုပ်ချင်တယ်ဆိုရင် null Return ပြန်ထားတာထက်စာရင် ပိုပြီးတော့ လွယ်လွယ်ကူကူ Check ပြုလုပ်လို့ရမှာဖြစ်ပါတယ်။

27. Copyright , Company Name , Description , Version Number စသည်တို့ရဲ့ Information တွေကိုစုစည်းသိမ်းထားချင်တယ်ဆိုရင် Assembly Info File ကို သုံးပါ။

28. ကိုယ်ရေးသားဖန်တီးထားတဲ့ File တွေကိုစနစ်တကျ Folders လေးတွေခွဲပြီးတော့သိမ်းထားပါ။ထိုသို့သိမ်းမယ်ဆိုရင်တော့ Hierarchies Level နှစ်ခုနဲ့သိမ်းနိုင်ပါတယ်။

ASP.Net မှာဆိုရင် ပိုသိသာပါတယ်။ဖိုင်တွေများလေ Folder လေးတွေဆောက်ပြီးတော့ခွဲရလေပါဘဲ။

Coding Standard အရ အများဆုံးအရေအတွက်ကတော့ Root Folder အထဲမှာစုစုပေါင်း 10 Folders အထိဘဲရှိသင့်ပါတယ်။အဲ့ဒီ Folders တစ်ခုချင်းစီမှာလည်း Sub Folders ပေါင်း 5 ခုအထိရှိလို့ရပါတယ်။အဲ့ဒီထက်များသွားတယ်ဆိုရင်တော့ Coding Standard အရ Hierarchies Level နှစ်ခုထဲကိုထည့်သွင်းလို့မရတော့ပါဘူး။ဒါကြောင့် Multiple Assembly တွေကိုထပ်ခွဲဖို့လိုလာတဲ့အတွက်ကြောင့် မလိုအပ်ဘဲနဲ့ ရှုပ်ထွေးစေနိုင်ပါတယ်။

29. အကယ်၍ ကိုယ်ရေးသားနေတဲ့ Code တွေက Database Connection နဲ့ပက်သက်တာ...ဒါမဟုတ် Sockets , FileStream ..... စတာတွေနဲ့ပက်သက်ပြီးတော့ အလုပ်လုပ်တော့မယ်ဆိုရင် ကိုယ်ရေးမဲ့ Data Access Code တွေက try{} finally{} နဲ့ ရေးသားထားဖို့တော့လိုပါလိမ့်မယ်။

ဘာကြောင့်လဲဆိုတော့ Database ကိုချိတ်ဆက်ရာမှာ Connection ကိုဖွင့်ရပိတ်ရတဲ့အတွက်ကြောင့် ဖွင့်ပြီးတော့မှ Error တက်ခဲ့ရင် မပိတ်မိရင်နောက်လူတွေသုံးရင် အဆင်မပြေဖြစ်တတ်တဲ့အတွက်ကြောင့် ဘာဖြစ်ဖြစ် နောက်ဆုံး Connection ကိုပြန်ပိတ်ဆိုပြီးတော့ finally Block ထဲမှာ ရေးသားထားဖို့လိုပါလိမ့်မယ်။

ပိုပြီးတော့ရှင်းသွားအောင် ဥပမာလေးနဲ့ကြည့်လိုက်ရအောင်။

```
Private void GetData()
{
        SqlConnection sqlcon=new SqlConnection();
        …..
        …..
        …..

        try
        {
                sqlcon.Open();
                ……
        }

        Finally
        {
                sqlcon.Close();
        }
```

```
}
```

ဆိုပြီးတော့ရေးသားထားဖို့လိုပါလိမ့်မယ်။ဒါမှသာ Error တက်ရင်လည်း Connection ကပိတ်သွားမှာဖြစ်ပါတယ်။

30. Variables တွေကိုကြေညာတော့မယ်ဆိုရင် ကိုယ်အရင်ဦးဆုံးသုံးမဲ့ Variables နဲ့တတ်နိုင်သမျှ အနီးကပ်ဆုံးထားပြီးတော့ ကြေညာသင့်ပါတယ်။နောက်တစ်ခုက Variables တွေကိုကြေညာရင် တစ်ကြောင်းကို တစ်ခုသာကြေညာသင့်ပါတယ်။

```
string name;
int age;
int phone;
```

အဲ့ဒီလိုမျိုးကြေညာသင့်ပါတယ်။

```
string name; int age; int phone;
```

အဲ့ဒီလိုမကြေညာသင့်ပါဘူး။

31. loop အထဲမှာရှိတဲ့ string objects တွေကို Manipulate လုပ်ဖို့လိုတယ်ဆိုရင်တော့ string အစား StringBuilder Class ကိုအသုံးပြုပါ။

ဘာကြောင့်လဲဆိုတော့ string objects က .Net မှာအလုပ်လုပ်ပုံကထူးဆန်းပါတယ်။

ဘယ်လိုမျိုးလဲဆိုတော့ တစ်ကြိမ်ကြေညာဖူးတဲ့ string ထဲကို တန်ဖိုးနောက်တစ်ကြိမ်ထည့်မယ်ဆိုရင် ရှိပြီးသား string object ကိုလွှင့်ပစ်လိုက်ပြီးတော့ နောက်ထပ် string object အသစ်တစ်ခုကိုထပ်ဆောက်ပါတယ်။မလိုအပ်ဘဲနဲ့ အမှိုက်တွေများလာနိုင်တဲ့အတွက်ကြောင့်ဖြစ်ပါတယ်။

ပိုပြီးရှင်းသွားအောင် Sample လေးနဲ့ကြည့်လိုက်ရအောင်။

```
public string ComposeMessage (string[] lines)
{
  string message = String.Empty;

  for (int i = 0; i < lines.Length; i++)
  {
    message += lines [i];
  }

  return message;
}
```

အပေါ်ကဥပမာလေးနဲ့ကြည့်မယ်ဆိုရင် string object message ထဲကို lines arrays တွေကို loop ပတ်ပြီး တော့ထည့်ထားတဲ့ပုံစံပါဘဲ။

ဒါပေမဲ့အမှန်တကယ်ဖြစ်နေတာကတော့ looping တစ်ကြိမ်ပတ်လိုက်တိုင်း ဆောက်ထားတဲ့ string object တွေကိုပျက်ပျက်သွားပြီးတော့ နောက်ထပ် အသစ်တစ်ခုအနေနဲ့ပြန်ပြီးတော့ recreation လုပ်ပါတယ်။အသစ်ပြန်ရမှ နောက်ထပ်တစ်ကြိမ်ပြန်ပြီးတော့ loop ပတ်ပြီးတော့ထပ်ထည့်ပါတယ်။

အဲ့ဒီလိုဖြစ်နေတဲ့အတွက်ကြောင့်အရင်ရှိပြီးသားအဟောင်းတွေကိုဖျက်ပစ်လိုက်ပြီးတော့ အသစ်တွေ ဆောက်ဆောက်တတ်တာကတော့ string object ရဲ့သဘောပဲဖြစ်ပါတယ်။

ဒါကြောင့် အဲ့ဒီလိုမဖြစ်စေရန်အတွက် loop အထဲမှာအလုပ်လုပ်ချင်တယ်ဆိုရင်တော့ string object အစား

StringBuilder class ကိုအသုံးပြုပါလို့ဆိုထားတာဖြစ်ပါတယ်။

ဒါကြောင့် Looping များများပတ်ပြီးတော့ တန်ဖိုးတွေထည့်ချင်တယ်ဆိုရင်တော့ string object အစား

StringBuilder class ကိုအသုံးပြုရင်တော့ Coding Standard ကိုလိုက်နာထားတဲ့အတွက်ကြောင့်

အသုံးပြုသင့်ပါတယ်။

ဒါဆိုရင် ကျွန်တော်တို့ပို့ရှင်းသွားအောင် StringBuilder class လေးနဲ့ကြည့်လိုက်ရအောင်။

```
public string ComposeMessage (string[] lines)
{
   StringBuilder message = new StringBuilder();

   for (int i = 0; i < lines.Length; i++)
   {
     message.Append( lines[i] );
   }

   return message.ToString();
}
```

StringBuilder ကိုသုံးထားတဲ့အတွက်ကြောင့် string object လိုမျိုး မဖြစ်တော့တဲ့အတွက်ကြောင့် Coding Standard အရသူ့ကိုသုံးသင့်ပါတယ်။

Looping အတွက်ကိုသာ StringBuilder ကိုအသုံးပြုသင့်ပါတယ်။

တခြားနေရာတွေမှာဆိုရင်တော့ StringBuilder အစား string ကိုသာအသုံးပြုရင် အကောင်းဆုံးဖြစ်ပါတယ်။

ဒါဆိုရင် ကျွန်တော့်တို့အခုလေ့လာနေတဲ့ Good Programming Practices တွေကတော့ဒီလောက်ပါဘဲ...။

ဒါဆိုရင် ကျွန်တော်တို့နောက်ထပ်လေ့လာမှာကတော့ Coding တွေရေးတဲ့အခါမှာ Layer တွေခွဲရေးသင့်တဲ့ အချက်လေးတွေ... Architecture အကြောင်းလေးတွေကို Coding Standard နဲ့အတူလေ့လာကြမှာဖြစ်ပါ တယ်။

အဲ့ဒါကတော့ Programmer တွေနဲ့သိပ်မစိမ်းလောက်ပါဘူး။

ဒါဆိုရင် ကျွန်တော်တို့ ဘယ်လိုလိုက်နာကြမလဲဆိုတာကို လေ့လာကြည့်လိုက်ရအောင်။

Architecture

Architecture မှာလိုက်နာသင့်တဲ့ အချက်လေးချက်ရှိပါတယ်။ဘာတွေလဲဆိုတော့

1. Layer ခွဲရေးတဲ့ ပုံစံကိုအမြဲအသုံးပြုသင့်ပါတယ်။
Multi Layer (n-Tier) Architecture ကိုအသုံးပြုသင့်ပါတယ်။

2. UI (User Interface) ကနေဘယ်တော့မှ Database ကို Access ပြုလုပ်တဲ့ Code တွေကိုမရေးသင့်ပါဘူး။

ASP.Net က SqlDataSource ကတော့ ခြွင်းချက်အနေနဲ့ သတ်မှတ်လို့ ရပါတယ်။

အမြဲတမ်း Data Access Layer ကိုရေးထားပြီးတော့ Database နဲ့ပက်သက်တဲ့ အလုပ်တွေကိုသူ့အထဲမှာရေးပါ။

အဲ့ဒီလိုပြုလုပ်ခြင်းကြောင့် နောင်တစ်ချိန်မှာ Database နဲ့သက်ဆိုင်တဲ့အကြောင်းအရာတွေကိုပြင်ချင်လို့ဘဲ

ဖြစ်ဖြစ် Update လုပ်ချင်လို့ဘဲဖြစ်ဖြစ်...ရွှေ့ပြောင်းချင်လို့ဘဲဖြစ်ဖြစ် အဆင်ပြေလွယ်ကူစွာပြုလုပ်နိုင်မှာဖြစ်ပါတယ်။

3. Database နဲ့သက်ဆိုင်တဲ့ Code တွေကိုရေးတော့မယ်ဆိုရင် Database Exception တွေကိုဖမ်းဖို့

အတွက် try catch finally Block တွေရေးဖို့မမေ့ပါနဲ့။အဲ့ဒီလို Exception Handler ရေးသားခြင်းက

Database မှာတက်တဲ့ Exception တွေရဲ့ Record ကိုပြေးမှာဖြစ်ပါတယ်။

Record တွေကိုပြေးတဲ့အခါမှာ အသေးစိတ်ပြေးသင့်ပါတယ်။ Command မှားနေတာလား...

Stored Procedure ကြောင့်လား... ConnectionString ကြောင့်လား...
စသဖြင့်အသေးစိတ်ပြေးသင့်ပါ တယ်။

အဲ့ဒီလို Record လုပ်ပြီးတဲ့အခါမှာ Exception rethrown လုပ်ပေးဖို့လိုပါလိမ့်မယ်။

ဒါမှသာ သူနဲ့ဆက်စပ်နေတဲ့ Application ထဲမှာရှိတဲ့ တခြား Layer တွေကို Exception catch

လုပ်ပြီးတော့ သင့်လျော်တဲ့ Action တွေကိုဆက်လက်လုပ်ဆောင်နိုင်မှာဖြစ်ပါတယ်။

4. နောက်ဆုံးတစ်ချက်ကတော့

Independent Utility Classes တွေကို Class Library တွေအဖြစ်သီးသန့်ခွဲထားပါ။

Database နဲ့သက်ဆိုင်တဲ့ ဖိုင်တွေအားလုံးကို နောက်ထပ် Class Library တစ်ခုအနေနဲ့ထားခြင်းဖြင့်

Coding Standard ကိုလိုက်နာပြီးတော့ ရေးသားသင့်ပါတယ်။

ဒါဆိုရင် ကျွန်တော်တို့နောက်ထပ်လေ့လာကြမှာကတော့ ASP.Net ရေးရင် Coding Standard ကိုလိုက်နာ

ပြီးတော့ ဘယ်လိုရေးရသလဲဆိုတာကို လေ့လာကြည့်လိုက်ရအောင်။

Some of ASP.Net Coding Standards

1. Session Variables တွေကို Code တွေကြားထဲမှာမသုံးပါနဲ့။

Session Variables ကို Class တွေထဲမှာဘဲအသုံးပြုသင့်ပါတယ်။

နောက်တစ်ခုကတော့ Session Variables ထဲမှာ Stored လုပ်ထားတဲ့ တန်ဖိုးတွေကို Access လုပ်တဲ့ Method တွေအထဲမှာဘဲအသုံးပြုသင့်ပါတယ်။

Session Class ကို Access လုပ်ချင်ရင်တော့

Using System.Web.HttpContext.Current.Session;

ကိုကြေညာပြီးတော့ Access လုပ်လို့ရပါတယ်။

2. Session ထဲမှာ Object အကြီးကြီးနဲ့ အများကြီးသိမ်းခြင်းကို မပြုလုပ်သင့်ပါဘူး...
ဘာကြောင့်လဲဆိုတော့ Session က Server Memory ပေါ်မှာသိမ်းတဲ့အတွက်ကြောင့်
User များလာတဲ့အမျှ Server ကြီးလေးလေးလာမှာဖြစ်ပါတယ်။

ဒါကြောင့် Session ထဲ မှာ Object အကြီးကြီးတွေ သိမ်းခြင်းကိုရှောင်ကြဉ်သင့်ပါတယ်။

3. Page တွေရဲ့ Look and Feel အတွက်ကတော့ Style Sheet (CSS) ကိုဘဲအသုံးပြုသင့်ပါတယ်။

Font-Names တို့ Font-Size တို့ကို ဘယ်တော့မှ UI Page(.aspx) မှာမကြေညာပါနဲ့။

အားလုံးကိုတစ်ခုထဲထားချင်တယ်ဆိုရင်တော့ Skin File ထဲမှာသွားကြေညာပြီးမှ Page
ကနေယူသုံးခြင်းကိုလည်း ပြုလုပ်သင့်ပါတယ်...။

ထိုသို့ပြုလုပ်ခြင်းကြောင့် နောင်တစ်ချိန်မှာ မိမိ Website ကို ပြောင်းလဲတဲ့အခါမှာ
အဆင်ပြေလွယ်ကူမှာဖြစ်ပါတယ်။

ဘာကြောင့်လဲဆိုတော့ Website ကမလှလို့ ဘဲဖြစ်စေ....မကြိုက်တော့လို့ ဘဲဖြစ်စေ...ပြင်မယ်ဆိုရင်
အားလုံးကို လိုက်ပြင်နေစရာမလိုတော့ဘဲ...CSS နဲ့ Skin File ကိုသွားပြင်လိုက်ရင် သူ့ တို့ ကိုယူသုံးထားတဲ့
Page တွေအားလုံးက အလိုအလျောက်ပြောင်းပေးသွားမှာဖြစ်တဲ့အတွက်ကြောင့်
များစွာအဆင်ပြေပါတယ်..။

နောက်တစ်ချက်က HTML Server Controls တွေကိုသုံးမယ်လို့ ယူဆပြီးတော့ စရေးတာနဲ့
...ကိုယ်ရေးနေတဲ့ Project တစ်ခုလုံးကို HTML Server Controls တွေနဲ့ ဘဲရေးသင့်ပါတယ်။

HTML Server Controls ကိုလည်းသုံးတယ်...ASP.Net Web Server Controls
တွေကိုလည်းသုံးတယ်...နှစ်မျိုးစလုံးကိုသုံးပြီးတော့ Project တစ်ခုထဲမှာမရေးသင့်ပါဘူး...။

တစ်မျိုးထဲကိုသာသုံးပြီးတော့ရေးသားသင့်ပါတယ်...။

Comments

Comment ပိတ်တဲ့အခါမှာလည်း လိုက်နာရမဲ့အချက်တွေက အများကြီးရှိပါတယ်။ဒါပေမဲ့ အသုံးများမဲ့...အသုံးတည့်မဲ့ အချက်လေးတွေကိုဘဲရွေးချယ်ဖော်ပြလိုက်ပါတယ်။

တော်ကြာ.." ဟ Comment ပိတ်တာတောင် ကိုယ့်စိတ်ကြိုက်မပိတ်ရဘူးလား " ဆိုပြီးဖြစ်လာမှာစိုးလို့ ပါ။ ☺

၁. Coding တစ်ကြောင်းရေး Comment တစ်ကြောင်းပိတ်ခြင်းကိုရှောင်ကြဉ်ပါ။
Variables ကြေညာပြီးတိုင်းလည်း Comment ပိတ်ခြင်းကိုရှောင်ကြဉ်ပါ။

၂. Comment ပိတ်တော့မယ်ဆိုရင် HTML မှာ Comment ပိတ်သလိုမျိုး မပိတ်ပါနဲ့။
/* .... */ ဆိုပြီးတော့မပိတ်ပါနဲ့။သူ့အစား // (or) /// ကိုသုံးပြီးတော့ပိတ်ပါ။
ASP.Net မှာဆိုရင်လည်း <%-- --%> ကိုသုံးပါ။

၃. နောက်ဆုံးတစ်ချက်တော့ တကယ် အသုံးလိုတဲ့ အခါမှသာ Comment ပိတ်ပါလို့အကြံပေးချင်ပါတယ်။
နောက်ဆုံးလေ့လာကြမှာကတော့ Exception ကို ဖမ်းတဲ့အခါမှာ ဘယ်လိုမျိုးကိုကျင့်သုံးလိုက်နာရမလဲဆိုတာကို လေ့လာကြမှာဖြစ်ပါတယ်။

Exception

1. Exception ဖမ်းတဲ့အခါမှာ catch ကိုရေးပြီးတော့ catch ထဲမှာ ဘာအလုပ်မှမလုပ်ခြင်းကိုရှောင်ပါ။

အကယ်၍ Exception မဖမ်းထားဘူးဆိုရင် ကိုယ်ရေးထားတဲ့ Code တွေက Exception တက်မတက်

ကိုမသိနိုင်တော့တဲ့အတွက်ကြောင့် ဘာရေးရေး Exception ဖမ်းတဲ့အကျင့်ကိုပြုလုပ်ပါ။

Developer အများစုကတော့ အရေးမကြီးတဲ့ Error တွေအတွက် လျစ်လျူရှုဖို့အတွက် ဤနည်းကို အသုံးပြုကြပါတယ်။

Programmatically Error တွေကို Checking လုပ်ခြင်းကိုရှောင်ကြဉ်သင့်ပါတယ်။

တစ်ချို့ကျတော့လည်း Exception တော့ဖမ်းထားပါရဲ့....ဒါပေမဲ့ ဘာမှမလုပ်ဘူး...ကုန်ကုန်ပြောရရင် Message တစ်ကြောင်းတောင်ရိုက်မပြတဲ့ လူများကိုလည်းတွေ့ဖူးပါတယ်။

အဲ့ဒီလိုပြုလုပ်ခြင်းကို ရှောင်ရှားသင့်ပါတယ်။

ဒါကြောင့် Exception ဖမ်းမယ်ဆိုရင်တော့ အနည်းဆုံး စာသားတစ်ကြောင်းလောက်တော့ရိုက်ပြသင့်ပါတယ်။

2. နောက်တစ်ချက်ကတော့ အပေါ်မှာလေ့လာခဲ့သလိုဘဲ ရင်းနှီးကျွမ်းဝင်တဲ့ Message တွေကိုဘဲရိုက်ပြပြီးတော့ User ကိုပြသင့်ပါတယ်။

3. Exception ဖမ်းရာမှာ Specific Exception တွေကိုသာဖမ်းပါ...။တစ်ချို့ကျတော့ catch ကိုဒီလိုရေးတတ်ပါတယ်။

```
catch(Exception)
{
        Console.WriteLine("ERROR");
}
```

အဲ့ဒီလိုမပြုလုပ်သင့်ပါဘူး။တိကျတဲ့ ပုံစံနဲ့ဖမ်းရမှာဖြစ်ပါတယ်။ဒီလိုပါ။

```
catch(FileIOException)
{
        Console.WriteLine(“IO Error”);

}
```

အဲ့ဒီလိုစစ်သင့်ပါတယ်။

ဒါပေမဲ့....အခြေအနေတစ်ရပ်တော့ရှိပါတယ်။ဘာလဲဆိုတော့ အခုမှစလေ့လာတဲ့ သူဖြစ်တယ်ဆိုရင်တော့ Specific Exception ကိုမသိသေးတဲ့အတွက်ကြောင့် Exception ဆိုပြီးတော့ ဖမ်းရင်လည်းဖြစ်ပါတယ်။

ဒါကြောင့် ကိုယ်က ဘယ် Exception ကိုဖမ်းမလဲဆိုတာကို သိရင်တော့ Specific Exception ကိုဖမ်းပါ။ မသိသေးဘူးဆိုရင်တော့ Exception လို့ သာဖမ်းပါ...အဲ့ဒီအတွက်တော့ ခြွင်းချက်လေး ပေးထားပါတယ်။

ဒီလောက်သိသွားပြီဆိုရင်တော့ ကျွန်တော်တို့လေ့လာနေတဲ့ Coding Standard အကြောင်းကတော့ ဒီမျှပါဘဲ။

ကျွန်တော်မေ့နေတာတို့..အရေးကြီးတဲ့အချက်လေးတွေကျန်ခဲ့တာတို့...အယူအဆမှားနေတာတို့....စသဖြင့် အမှားတွေရှိနေခဲ့တယ်ဆိုရင် လည်း ခွင့်လွှတ်ပေးပါလို့တောင်းပန်ပါတယ်။

ဘာဖြစ်လို့ကျွန်တော်တို့တွေ အချိန်ကုန်ခံပြီးတော့ Coding Standard တွေကိုလေ့လာနေကြသလဲ။

ဘာဖြစ်လို့ ကျွန်တော်က ကော စာဖတ်သူတို့ အတွက် အချိန်ကုန်ခံပြီးတော့ ဒီစာတွေရေးပေးနေသလဲ ...။

ဘာကြောင့်လဲဆိုတော့ ကျွန်တော့်သဘောကတော့ စာဖတ်သူတို့ ကိုစနစ်တကျ ဖြစ်စေချင်တဲ့အတွက်ကြောင့်သာဖြစ်ပါတယ်။

ကိုယ်ရေးနေကြအတိုင်း ရေးရင်ကော မရဘူးလား။

အဲ့ဒီလို လိုက်ပြီးတော့ အားလုံးကိုလိုက်နာနေမယ်ဆိုရင် ဘယ်မှာလာပြီးတော့ Program ကပြီးတော့မှာလဲ...Deadline မဝင်ရင် Project Manager ကဆူလိမ့်မယ်။

အကုန်လုံးကိုလိုက်နာနေရင်တော့ အချိန်ကုန်ပြီးတော့ ခရီးရောက်မှာမဟုတ်ဘူး.....စသဖြင့်ပြောလို့ရပါတယ်။

ဟုတ်ပါတယ်။အားလုံးကိုလိုက်နာရမယ်လို့မဆိုလိုပါဘူး။ဒါဆိုရင် ဘာကြောင့် လေ့လာနေရသလဲ...

ကျွန်တော့် ဆရာတစ်ယောက်ပြောဖူးတာကတော့ သူအလုပ်လုပ်ဖို့ အင်တာဗျူးသွားဖြေတုန်းက သူ့ကိုမေးတယ်...မင်းတစ်နေ့ကို Coding Line ရေဘယ်လောက်ရေးနိုင်လဲဆိုပြီးတော့မေးပါတယ်။

အဲ့ဒီတုန်းကသူလည်းမသိဘူးလို့ပြန်ဖြေခဲ့ပါတယ်။အဲ့ဒီအချက်တွေကလည်းအရေးကြီးပါတယ်။

ဘာဖြစ်လို့လဲဆိုတော့ သူတို့ဆီမှာက Project တစ်ခုကိုရေးရင် တစ်ရက်ကိုလိုင်းရေဘယ်လောက်ရေးနိုင်သလဲဆိုတာကိုကြည့်ပါတယ်...။

ပြီးရင် သူတို့ရေးနိုင်တဲ့လူအရေအတွက်ကိုခေါ်ပါတယ်။

ဥပမာ တစ်ယောက်က တစ်ရက်ကို ၂၀၀ လောက်ရေးနိုင်ရင် လိုင်းရေ ၁၀၀၀ ရှိတဲ့ Program ကိုရေးမယ်ဆိုရင် တစ်ရက်ထဲနဲ့ပြီးချင်ရင် ၅ ယောက်လောက်နဲ့ရေးခိုင်းပြီးတော့ ကျန်တဲ့ Programmer တွေကို တခြားအလုပ်တွေလုပ်ခိုင်းမှာဖြစ်ပါတယ်။

ဒါဆိုရင် အင်တာဗျူးတစ်ခုအတွက်နဲ့ဘဲ ဒါတွေကိုအချိန်ကုန်ခံပြီးတော့ လေ့လာနေရတာလား...ဆိုတော့မဟုတ်ပါဘူး။

အလုပ်ထဲမှာတောင် ကိုယ်နဲ့ အရည်အချင်း ချင်းတူတဲ့ လူ နဲ့ ယှဉ်ကြည့်ရင် သူလည်း ဒီ Program ကိုရေးနိုင်တယ်...ကိုယ်လည်းရေးနိုင်တယ်...ဒါပေမဲ့....Boss နေရာကကြည့်ရင် Coding Standard ကိုလိုက်နာပြီးတော့ စနစ်တကျ ရေးထားတဲ့သူဆိုရင် ပိုပြီးတော့ အကြိုက်တွေ့ ပါတယ်...။

ဒါကြောင့် အဲ့ဒီလူက ရာထူးတက်ဖို့ ... လစာတိုးဖို့ အတွက်ပိုပြီးတော့ အခွင့်အရေးကောင်းမှာဖြစ်ပါတယ်။

နောက်ထပ် အကြောင်းပြစရာတော့ အများကြီးရှိပါသေးတယ်...ဒါပေမဲ့ ဒီလောက်ဘဲပြတော့မယ်။
နောက်ဆုံးအနေနဲ့ပြောရရင် ကျွန်တော် အခုရေးထားတဲ့ အကြောင်းအရာတွေကိုဖတ်ပြီးတော့ လိုက်နာနိုင်တယ်...မလိုက်နာနိုင်ဘူးဆိုတာကတော့ မိမိ ဘာသာ ဆုံးဖြတ်ပါ။
ဒါကတော့ စာဖတ်သူတို့ ရဲ့ ရွေးချယ်မှုပါဘဲ...ကျွန်တော်က အတင်းအကြပ် လိုက်ကို လိုက်နာရမယ်လို့ မပြောပါဘူးခင်ဗျာ...။

ကျွန်တော်ကတော့ အဆင်ပြေစေရန်အတွက် လိုက်နာချင်တဲ့သူတွေအတွက် လေ့လာရာမှာ အဆင်ပြေချောမွေ့စေရန်အတွက် သာရည်ရွယ်ပြီးရေးထားခြင်းဖြစ်တဲ့အတွက်ကြောင့် ကျွန်တော့်ရဲ့ C# Coding Standard များအကြောင်းကို ဒီနေရာမှာဘဲရပ်နားခွင့်ပြုပါခင်ဗျာ...။
အားလုံးကျန်းမာချမ်းသာကြပါစေ လို့ဆုမွန်ကောင်းတောင်းလိုက်ပါတယ်။

အစဉ်လေးစားစွာဖြင့်

Zawminsoel (Initial)

(MCTS , MCPD [Web])

(MCTS [MSSQL 2008])