

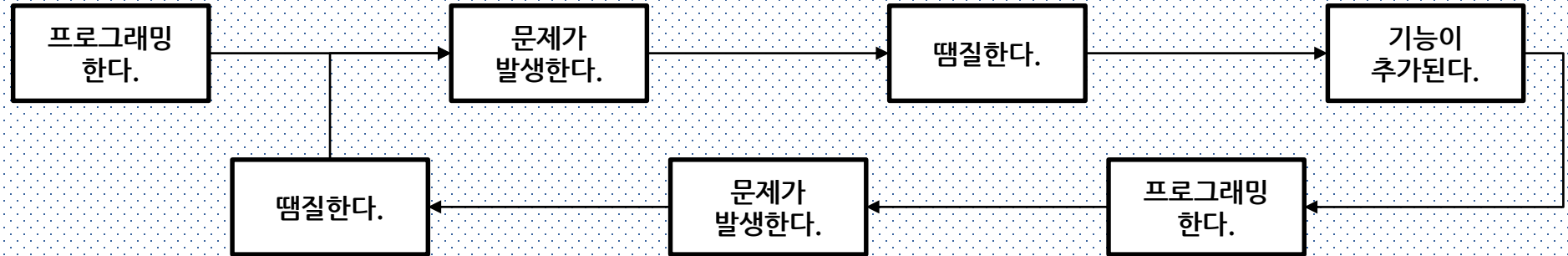
구현방법론

코드품질의 개요

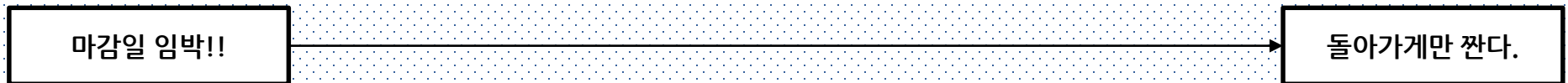
- 코드품질 확보의 필요성
- 코드품질 확보를 위한 활동 설명

코드품질 확보가 왜 필요한가?

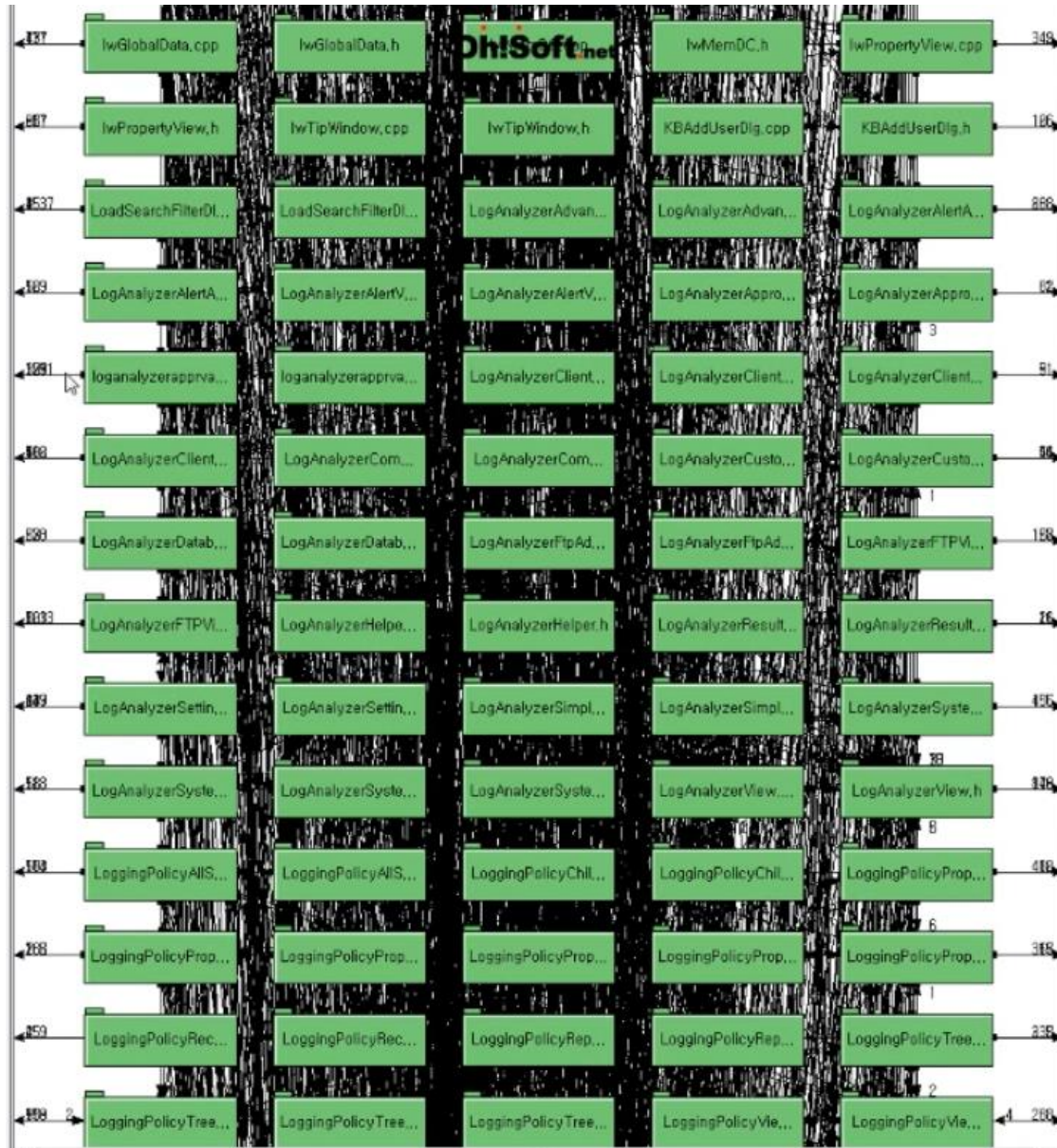
Case 1



Case 2



코드품질 확보가 왜 필요한가?



다수의 스파게티 코드



코드 수정의 어려움

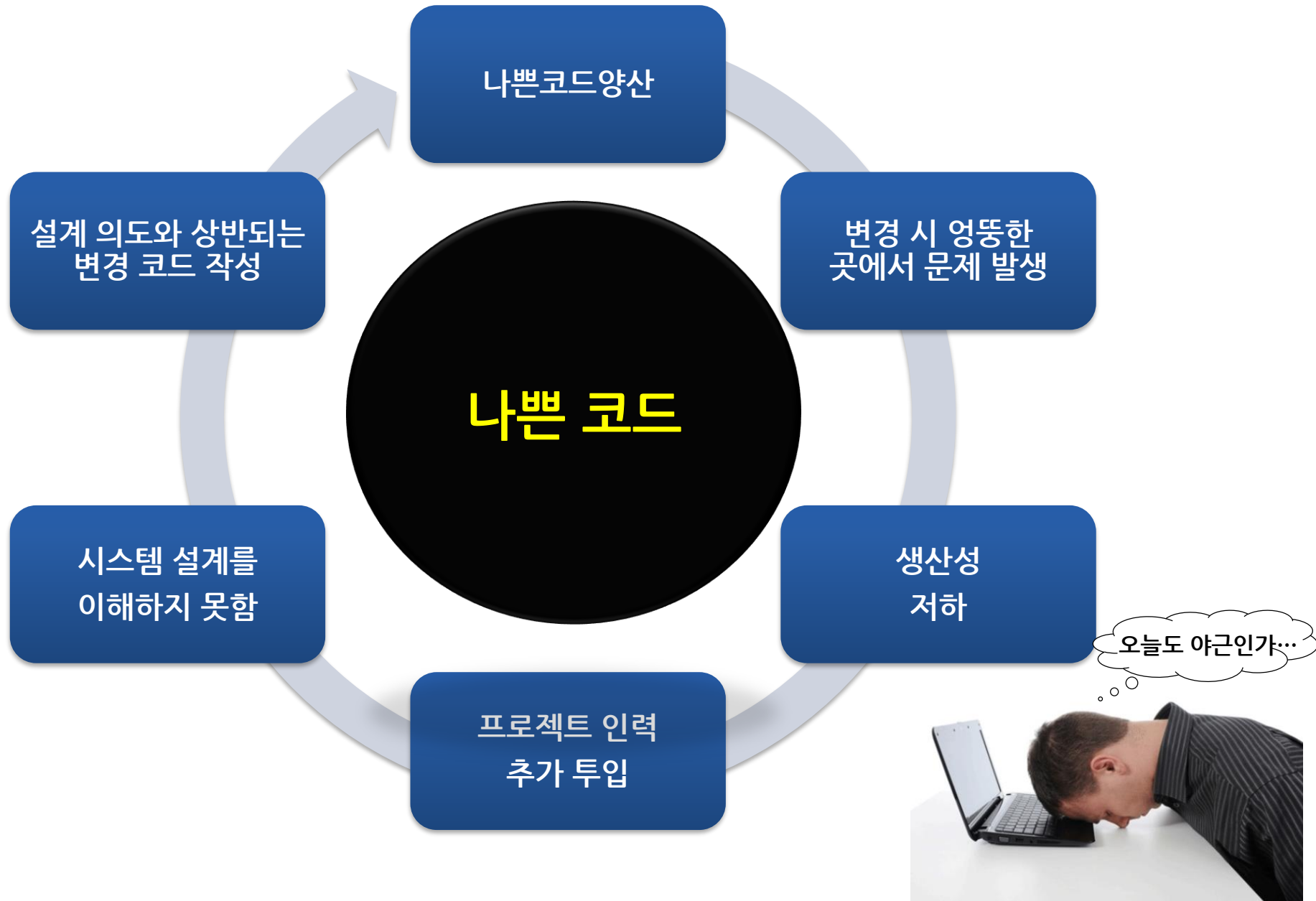


미흡한 영향평가로 새로운 버그 등장

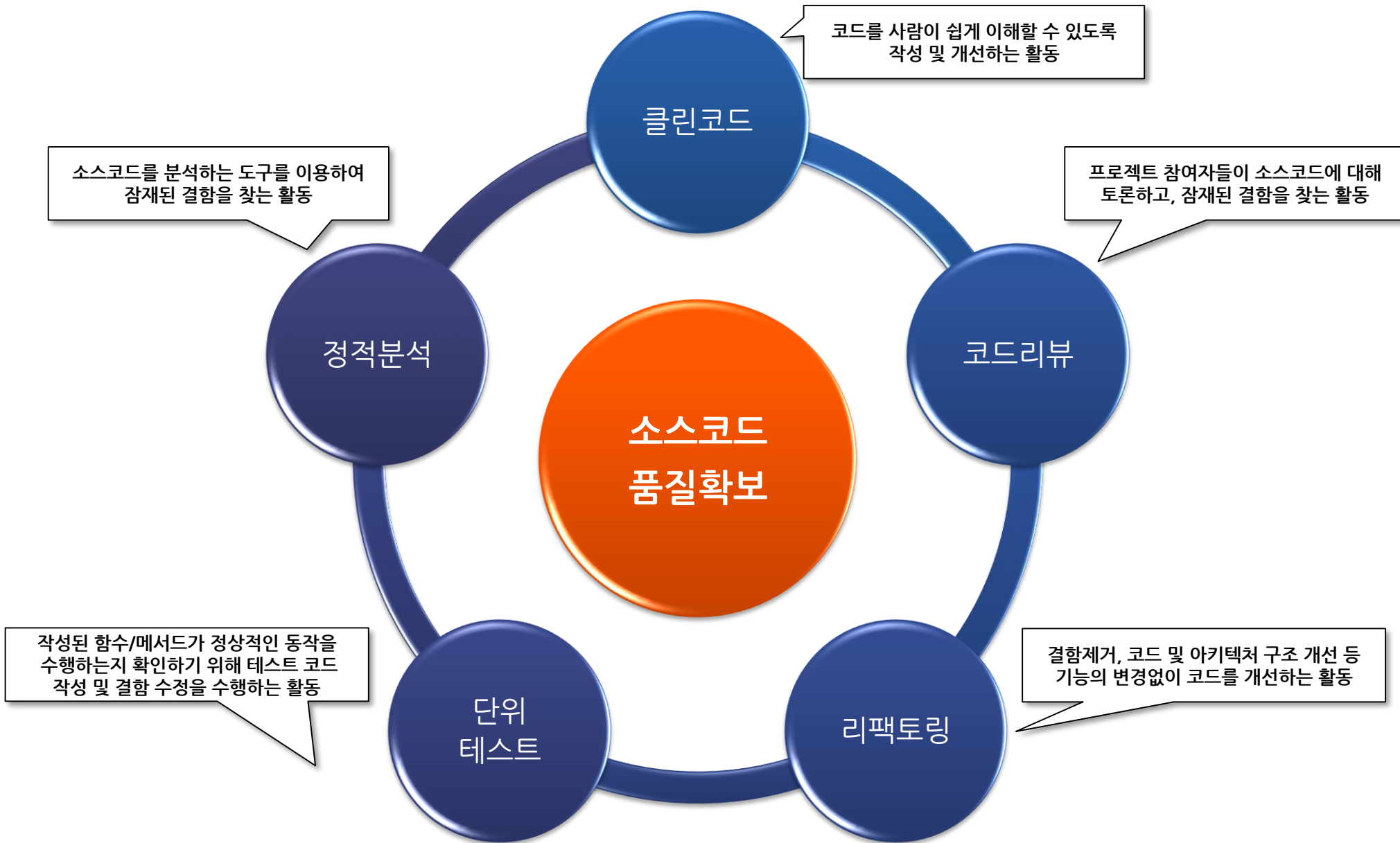


코드 수정 비용 증가

최종 제품의 품질을 보장하기 어려움



소스코드의 품질을 확보하는 활동은 무엇이 있을까요?



클린코드(Clean Code)

사람이 쉽게 이해할 수 있도록 코드를 작성하고 지속적으로 개선하는 활동

```
public class Car{
    private Tire tire1 = new Tire();
    private Tire tire2 = new Tire();
    private Tire tire3 = new Tire();
    private Tire tire4 = new Tire();
}
```

```
public class Car{
    private Tire leftFrontTire = new Tire();
    private Tire rightFrontTire = new Tire();
    private Tire leftBackTire = new Tire();
    private Tire rightBackTire = new Tire();
}
```

```
public class Vehicle{
    public moveControl(int moveType) {
        if (moveType==0) {

        } else if (moveType==1) {

        } else if (moveType==2) {

        } else {
            throw Exception();
        }
    }
}
```

개
선

```
public class Vehicle{
    Const int GO_STRAIGHT = 0;
    Const int TURN_LEFT = 1;
    Const int TURN_RIGHT = 2;
    Const int STOP = 3;

    public moveControl(int moveType) {
        if (moveType==GO_STRAIGHT) {

        } else if (moveType== TURN_LEFT) {

        } else if (moveType== TURN_RIGHT) {

        } else {
            throw Exception();
        }
    }
}
```

코드 리뷰(Code Review)

개발자에 의해 작성된 코드를 다른 개발자들의 다양한 관점과 시선에서 코드의 문제점을 파악하고 개선하는 활동

- 코드 리뷰 수행 시 이점
 - 코드품질 ↑, 결함률 ↓
 - 지식을 공유하고 공통의 코딩 가이드라인을 구축할 수 있다.
- 코드 리뷰를 잘 하려면
 - 코드 리뷰 중에는 **상냥**하게... (비판보다는 건설적인 논평)
 - 매 주 **정기적인** 코드 리뷰 시간을 할애하라. (최대 두 시간 정도)
 - 모든 리뷰 미팅 시 **구성원들의 역할을 변경**하라. (팀 구성원들 사이에 조직 서열의식을 갖지 않도록)
 - **신입사원**을 포함시켜라. (배운지 얼마 되지 않은 신선한 학교의 지식은 또 다른 지식을 제공할 수 있다.)
 - 경험과 지식을 가진 **전문가**를 포함시켜라.
(에러 코드를 더 빨리, 더 정확하게 검출할 수 있다.)



[샘플] Crucible을 이용한 코드리뷰

리팩토링(Refactoring)

소프트웨어를 보다 쉽게 이해할 수 있고, 적은 비용으로 수정할 수 있도록 겉으로 보이는 동작의 변화 없이 내부 구조를 변경하는 활동

- 리팩토링의 이점
 - 소프트웨어를 **더 이해하기 쉽게** 만들 수 있다.
 - 프로그램을 **빨리 작성**하도록 도와준다.
- 리팩토링 수행 시점
 - 틈틈이** 계속!
 - 삼진 규칙** (세 번째로 비슷한 것을 하게 되면)
 - 기능을 추가할 때, 버그를 수정할 때**
 - 코드 리뷰** 시
- 리팩토링 수행 시 유의해야할 사항은
 - 동작의 변경 여부를 **검증**하라. (테스팅! 테스트! 테스트!)
 - 현재 코드가 동작하지 않는다면, 코드를 다시 짜는게 나을 정도라면, → 리팩토링 하지 마라.**

```
public int getCharge(int quantity) {
    int charge = 0;
    Date now = new Date();

    // 현재 Summer타임 기간이라면
    if (now.before(SUMMER_START) || now.after(SUMMER_END)) {
        charge = quantity * WINTER_RATE +
                WINTER_SERVICECHARGE;
    } else {
        charge = quantity * SUMMER_RATE;
    }
    return charge;
}
```

개선

```
public int getCharge(int quantity) {
    int charge = 0;
    Date now = new Date();

    if (isSummerTime(now)) {
        charge = quantity * WINTER_RATE +
                WINTER_SERVICECHARGE;
    } else {
        charge = quantity * SUMMER_RATE;
    }
    return charge;
}

private boolean isSummerTime(Date now) {
    return now.before(SUMMER_START) || now.after(SUMMER_END);
}
```

단위테스트(Unit Test)

프로그램의 각 모듈을 고립시켜서 각각의 모듈이 의도한 대로 정확하게 동작하는지 확인하는 테스트

- 단위 테스트의 목적은
 - 프로그램의 각 모듈을 고립시켜서 **각각의 모듈이 의도한 대로** 정확하게 동작하는지 확인하는 것
- 단위 테스트의 대상(SUT)은
 - **개발자가 작성한 특정 모듈(함수 또는 클래스)에 대한 테스트**
- 단위 테스트의 수행 주체는 **개발자**.
- 단위 테스트는 **White Box** 테스트 기법이다.
 - White box 테스트? 프로그램의 로직을 이해하고, 내부 구조와 동작을 검사하는 소프트웨어 테스트 방식
- 단위 테스트의 Reader(독자)는
 - 코드 형태로 **개발자 만** 읽을 수 있음
- 단위 테스트는 XP(eXtreme Programming) 개발방법론의 핵심 실천방법
 - XP의 핵심 실천 방법인 TDD(Test-Driven Development)의 핵심 활동으로 단위 테스트 활용



소프트웨어로부터 만들어진 프로그램을 실제로 실행해보지 않고 코드를 분석하고, 개선의 방향을 파악하는 활동

- 정적분석 정의
 - **자동화된 Tool**을 이용하여 **코드를 분석**하고 개선의 방향(잠재 결함, 코드복잡도)등의 정보를 파악하는 과정
- 틀에 의한 정적분석 특징
 - 리뷰와 마찬가지로 장애(Failures)*보다는 **결함(Defects)** 발견**
 - 사전에 정의된 규칙이나 코딩 표준을 기반으로 분석함 (Rule Set의 최적화 필요)
- 정적분석 수행 시 이점
 - 테스트 실행 전에 **조기 결함 발견**
 - 높은 복잡도 측정치와 같은 메트릭을 계산하여 코드와 설계의 의심스러운 양상에 대한 **조기 경보**
 - 코드와 설계의 **유지보수성 향상**
- 정적분석 도구
 - 무료 : PMD, CPPCheck, FindBug ...
 - 상용 : Understand, Prevent, Ndepend, Klocwork ...

* 장애(Failures) : 결함(Defect)로 인한 의도되지 않은 동작 또는 결과

** 결함(Defect) : 시스템의 성능을 떨어뜨리거나 멈춤, 오동작을 유발하는 소프트웨어 구문

Clean Code

- By 비야네 스트롭스트롭 (C++의 창시자)
 - 나는 우아하고 효율적인 코드를 좋아한다. 논리가 간단해야 버그가 숨어들지 못한다. 의존성을 최대한 줄여야 유지보수가 쉬워진다. 오류는 명백한 전략에 의거해 철저히 처리한다. 성능을 최적으로 유지해야 사람들이 원칙 없는 최적화로 코드를 망치려는 유혹에 빠지지 않는다. 클린 코드는 한 가지를 제대로 한다.
- By 그래디 부치 (Object Oriented Analysis and Design with Application의 저자)
 - 클린 코드는 단순하고 직접적이다. 클린 코드는 잘 쓴 문장처럼 읽힌다. 클린 코드는 결코 설계자의 의도를 숨기지 않는다. 오히려 명쾌한 추상화와 단순한 제어문으로 가득하다.

- 데이브 토마스 (이클립스 전략의 대부)
 - 클린코드는 작성자가 아닌 사람도 읽기 쉽고 고치기 쉽다. 의미 있는 이름이다. 특정한 목적을 달성하는 방법은 하나만 제공한다. 의존성은 최소이며 각 의존성을 명확히 정의한다. API는 명확하며 최소로 줄였다. 때로는 정보 전부를 코드만으로 명확하게 드러내기 어려우므로 언어에 따라 문학적 표현이 필요하다.
- 존 제프리 (Extreme Programming Installed의 저자)
 - 중요한 순으로 나열하자면 간단한 코드는
 - 중복이 없다
 - 표현성이 뛰어나다
 - 작게 추상화되어 있다

컴퓨터가 이해하는 코드는 어느 바보나 다 짤 수 있다.

훌륭한 프로그래머는 사람이 이해할 수 있는 코드를 짤다

- 마틴파울러-

사람이 이해할 수 있는 코드란 무엇일까요?

- 가독성이 뛰어나다.
- 간단하고 작다.
- 의존성을 최대한 줄였다.
- 의도와 목적이 명확한 코드
- 타인에 의해 변경이 용이 한 코드
- 중복이 없는 코드
- 개체(Class, Method)가 한가지 작업만 수행하는 코드

왜 사람이 이해하는 코드를 만들어야 하나요 ?

Enhanced

Modified

Adapted

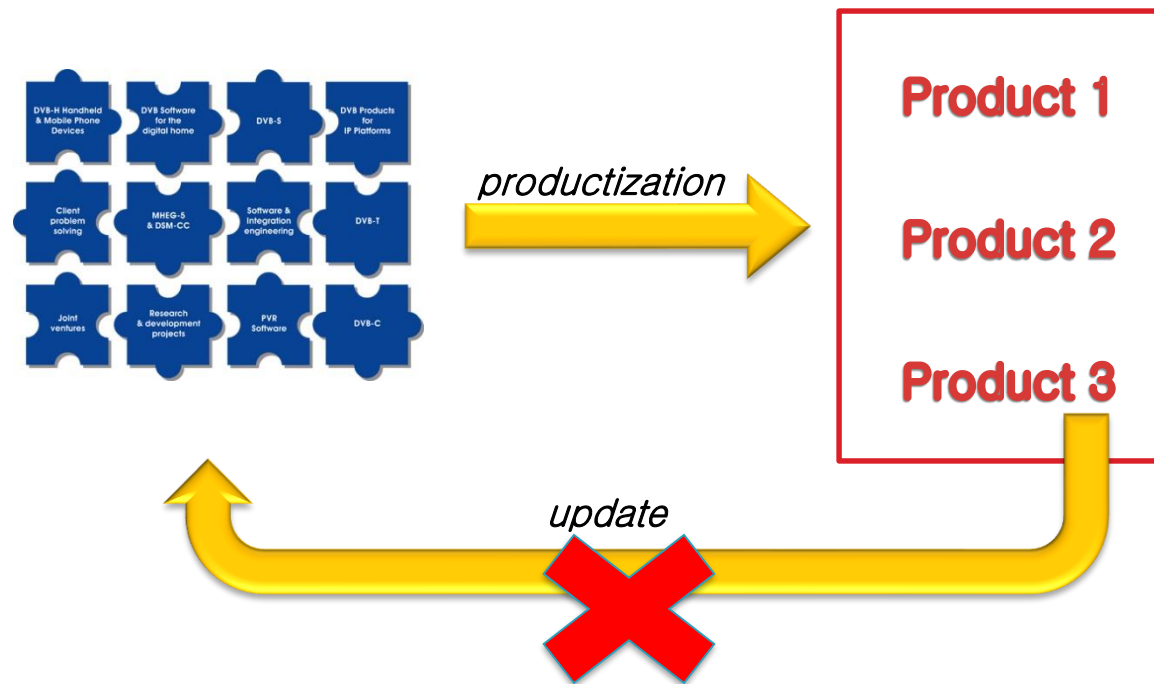


Evolution

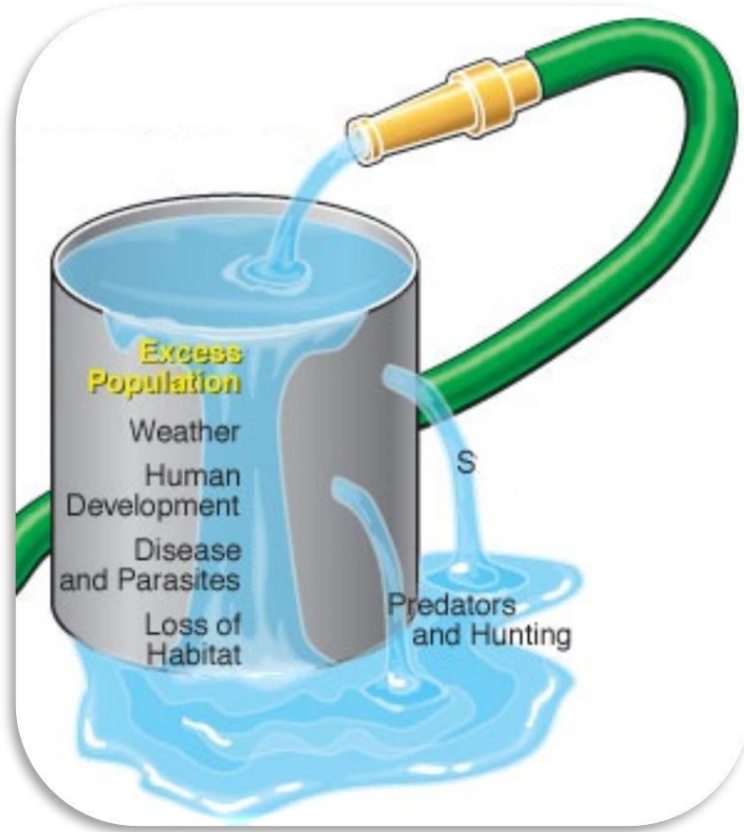
More complex

Drift away
from its original design

기능만 추가할 뿐 코드 품질의 개선 작업은 미비합니다



개선되지 않은 SW는 늙게 되어 Dirty code 가 됩니다



Dirty code는 SW Aging을 가속화
The Spiral of Complex

Symptom	Description	Metrics
Pollution	the system includes many components not necessary to carry out the business functions	Unused data, Duplicated code
No documents	the knowledge of the application domain and its evolution is spread over the programs and can no longer be derived from the documentation	Design spec, API doc, References
Poor lexicons	the names of components have little lexical meaning or are in any case inconsistent with the meaning of the components they identify	Naming convention
Tightly Coupling	the programs and their components are linked by an extensive network of data or control flows	Call graph, Fan-in, Fan-out
Architecture Erosion	the system's architecture consists of several different solutions that can no longer be distinguished; even though the software started out with a high quality basic architecture, the superimposition of these other hacked solutions during maintenance has damaged its quality	Dependency

왜 사람이 이해하는 코드를 만들어야 하나요 ?

- 코드가 건강합니다.
- 코드가 오래 갈 수 있습니다.
- 조직의 경쟁력을 높입니다.
- 개발 비용을 줄이고 생산성을 높입니다.

의미 있는 이름

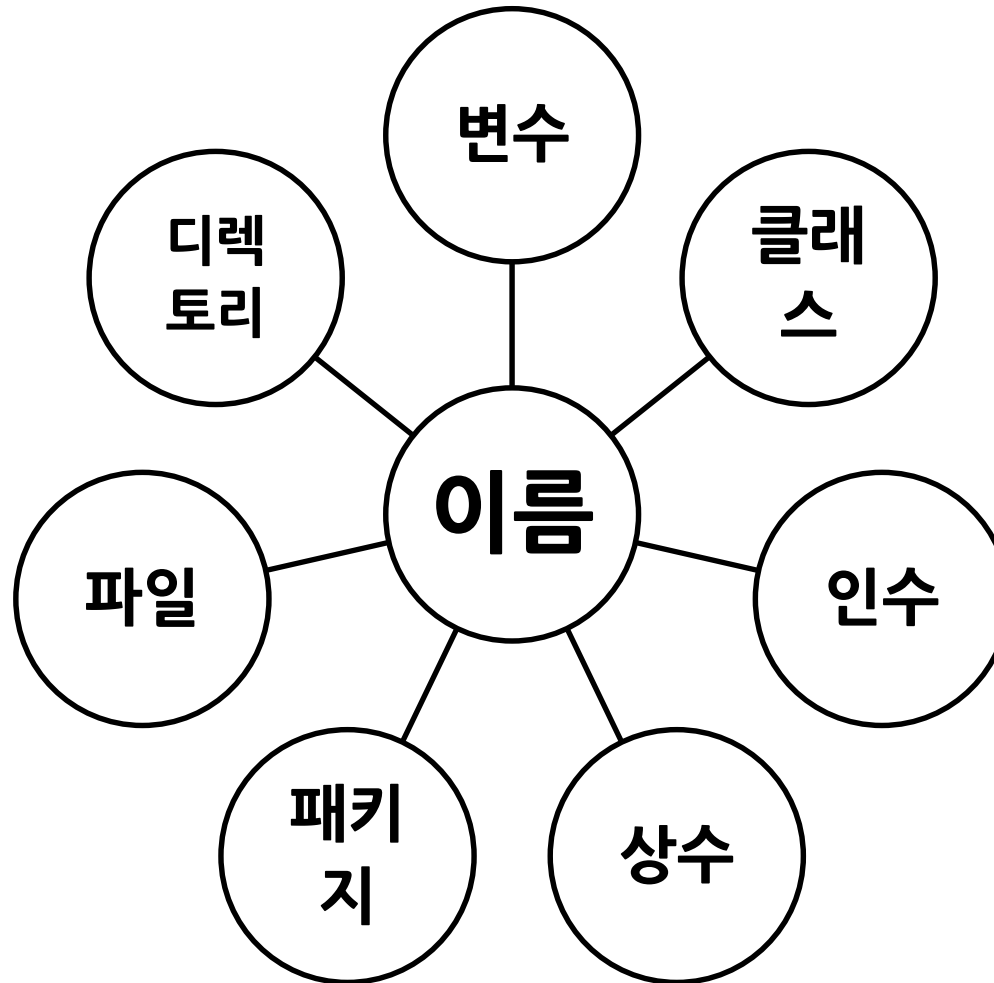
착한 함수

없는 게 나은 주석

형식이 맞는 코드

오류처리

이름 짓기가 가장 중요합니다.



주석이 필요 없는 코드를 작성하라

```
//Login Client으로부터의 Login요청을 처리함.  
void CMessageProcFunc::Net_CLIENT_INFO()
```

```
void CMessageProcFunc::processLoginRequestFromClient()
```

```
public List<int[]> getList(){  
    List<int[]> list1 = new ArrayList<int []>();  
  
    for(int[] x : theList) {  
        if(x[0] == 4) {  
            list1.add(x);  
        }  
    }  
  
    return list1;  
}
```


```
public List<Cell> getFlaggedCell(){  
    List<Cell> flaggedCells = new ArrayList<Cell>();  
  
    for(Cell cell : gameBoard){  
        if(cell.isFlagged()){  
            flaggedCells.add(cell);  
        }  
    }  
  
    return flaggedCell;  
}
```

그릇된 정보를 피하라

길고 서로 흡사한 이름은 피하라

```
public Class Vehicle {  
  
    controllForEfficientHandlingOfString ();  
    controllForEfficientStorageOfString ();  
  
}
```

```
public Class Y {  
    private Vehicle vehicle = new Vehicle();  
  
    public void method A(){  
        vehicle.controllForEfficientStorageOfString();  
    }  
}
```



IDE 자동완성기능에서 선택
할 때 실수의 여지가 있음

그릇된 정보를 피하라

연속적인 숫자를 붙인 이름을 피하라

```
public class Car{
    private Tire tire1 = new Tire();
    private Tire tire2 = new Tire();
    private Tire tire3 = new Tire();
    private Tire tire4 = new Tire();
}
```

```
public class Vehicle{

    public moveControl(int moveType){
        if(moveType==0){
        }else if(moveType==1){
        } else if(moveType==2){
        }else{
            throw Exception();
        }
    }
}
```

```
public class Car{
    private Tire leftFrontTire = new Tire();
    private Tire rightFrontTire = new Tire();
    private Tire leftBackTire = new Tire();
    private Tire rightBackTire = new Tire();
}
```

```
public class Vehicle{
    Const int GO_STRAIGHT = 0;
    Const int TURN_LEFT = 1;
    Const int TURN_RIGHT = 2;
    Const int STOP = 3;

    public controlMovement(int moveType){
        if(moveType== GO_STRAIGHT){
        }else if(moveType== TURN_LEFT){
        } else if(moveType== TURN_RIGHT){
        }else{
            throw Exception();
        }
    }
}
```

문맥에 맞는 단어를 사용하라

이해하는데 시간을 소모하지 않기 위해 일관성 있는 단어를 사용하라

```
public class Vehicle{  
  
    public double getFuelRemaining();  
    public double retrieveDegreeOfTireAbrasion();  
    public double fetchCurrentSpeed();  
  
}
```

```
public class Vehicle{  
  
    public double getFuelRemaining();  
    public double getDegreeOfTireAbrasion();  
    public double getCurrentSpeed();  
  
}
```

클래스 이름을 일관성 있게 작성하라

클래스 이름은 명사 또는 명사구로 작성하라

: 클래스는 행위(멤버함수)의 주체로서 명사 또는 명사구로 표현됨

클래스	변경된 클래스 이름
CBandWidthControl	CBandWidth
CFileControl	CStationFile
CIOCPServer	CStationServer
CIOCPServerDlg	CStationServer
CQuotaCheckThread	CQuotaChecker
CSearchThread	CFileFinder
CServerControl	CStationServerController

함수 이름을 일관성 있게 작성하라

함수 이름은 동사 또는 동사구로 작성하라

: 함수는 클래스가 행하는 행위로서 동사 또는 동사구로 표현됨

SRP Test: *클래스가 스스로 멤버함수 한다.*

클래스	멤버함수
CIOCPServer	startService
CIOCPServerDlg	startStationServer
CIOCPServerDlg	makeLogPath
CIOCPServerDlg	writeLog
CMessageProcFunc	getSharedQuota
CMessageProcFunc	isAvailableToAddPhysicalDiskQuota
CMessageProcFunc	getUserPath
CMessageProcFunc	getAppendedUserPath
CMessageProcFunc	isAvailableToAddSharedQuota
CMessageProcFunc	getUserPath
CMessageProcFunc	increaseSharedQuota
CMessageProcFunc	parseDownloadFileList

착한 함수

작게 만들어라

이상적인 블록(if문 While문)안의 내용은 한 줄
블록 안에서 다른 함수를 호출하도록 작성

```
public String renderPageWithSetupsAndTeardowns(PageData pageData, boolean isSuite){
```

```
    If(isTestPage) {
        WikiPage testPage = pageData.getWikiPage();
        StringBuffer newPageContent = new StringBuffer();
        includeSetupPages(testPage, newPageContent, isSuite);
        newPageContent.append(pageData.getContent());
        ...
    }
    return pageData.getHTML();
}
```

```
public String renderPageWithSetupsAndTeardowns(PageData pageData, boolean isSuite){
```

```
    If(isTestPage) {
        includeSetupAndTeardownPages(pageData, isSuite);
    }
    return pageData.getHTML();
```

```
}
```


한가지만 하도록 만들어라

동일 레벨로 추상화된 작업만 하도록 만들어라

한 가지란?

• 함수가 지정된 함수 이름 아래서 추상화 수준이 하나인 단계만 수행한다면 그 함수는 한가지 작업만 한다.

추상화 수준이
한가지 인 것은?

- 높음
 - getHTML();
- 중간
 - String pagePathName = PathParser.render(pagePath);
- 낮음
 - append("\n");

인수를 적게 만들어라

이상적인 인수개수는 0개이고 그 다음은 1개이다.

3개 이상의 인수가 필요하다면 객체 사용을 고려해라

```
public void drawCircle(double a, double b, double radius);
```

```
public void drawCircle(Point location, double radius);
```

```
public class Point{  
    double x;  
    double y;  
  
    double getX(){  
        return x;  
    }  
  
    double getY(){  
        return y;  
    }  
}
```

변경 시 여러 부분을 손대야 한다 오류가 발생할 확률이 높아진다

SearchThread 1162 ~

```
//QuataFree
stOut.ullQuataFree = 0;//youyou2_ULONGLONG

//File Name
_stprintf((LPTSTR)stOut.szName, TEXT("%s"), files.GetFileName());

//File Path
_stprintf((LPTSTR)stOut.szPath, TEXT("%s"), files.GetFilePath());

//File Path Rebuild Start
szFilePath.Format(_T("%s"),stOut.szPath);
nPoint = szFilePath.ReverseFind(_T('\\\\'));
szFilePath = szFilePath.Left(nPoint);
```

SearchThread 1314 ~

```
//QuataFree
stOut.ullQuataFree = 0;//youyou2_ULONGLONG

//File Name
_stprintf((LPTSTR)stOut.szName, TEXT("%s"), files.GetFileName());

//File Path
_stprintf((LPTSTR)stOut.szPath, TEXT("%s"), files.GetFilePath());

//File Path Rebuild Start
szFilePath.Format(_T("%s"),stOut.szPath);
nPoint = szFilePath.ReverseFind(_T('\\\\'));
szFilePath = szFilePath.Left(nPoint);
```



주석대신 코드로 의도를 표현해라

코드만으로는 의도를 표현하기 힘들 때만 주석을 사용해라
그러나 될 수 있으면 코드로 표현해라

//플래그된 셀을 반환

```
public List<int[]> getList(){
    //플래그된 셀을 List형태로 저장
    List<int[]> list1 = new ArrayList<int []>();

    for(int[] x : theList) {
        if(x[0] == 4) { //플래그 되어있다면
            list1.add(x); // 리스트에 저장
        }
    }

    return list1;
}
```

```
public List<Cell> getFlaggedCell(){
    List<Cell> flaggedCells = new ArrayList<Cell>();

    for(Cell cell : gameBoard){
        if(cell.isFlagged()){

            flaggedCells.add(cell);
        }
    }

    return flaggedCell;
}
```

명쾌한 의도파악이 힘든 주석을 피하라

제 3자가 이해하기 힘든 주석은 피해야 한다.

//용량 변경 시??

```
void CMessageProcFunc::Net_FILE_LIST_CHQUOTA(LPPER_HANDLE_DATA si, char *pBuf, int len)
{
    Net_FILE_LIST_CHDIR(si, pBuf, len);
}
```

```
if ( cType == (BYTE)(-1))
{
    cType = LIST_MPC; //youyou_MODIFY 맞나??
}
```

//BYTE -> TCHAR 변환하므로 TCHAR의 Max값은 BYTE의 반이다.(BYTE:1byte, TCHAR:2byte)그리고 마지막에 NULL이 들어 가는것에 대비해 +1해준다.

//PATHSIZE*3 을 반으로 나누기가 애매해서 PATHSIZE*2으로 계산 했음.

```
TCHAR szDir[PATHSIZE*2+1]; ::ZeroMemory(szDir, sizeof(szDir));
TCHAR szName[PATHSIZE*2+1]; ::ZeroMemory(szName, sizeof(szName));
TCHAR szNew[PATHSIZE*2+1]; ::ZeroMemory(szNew, sizeof(szNew));
```

이런 주석은 삭제해야 합니다

없어도 되는 주석

```
/**  
 * 달 중 날짜를 반환한다.  
 *  
 * @return 달 중 날짜  
 */
```

```
Public int getDayOfMonth(){ //달 중 날짜  
    private int dayOfMonth;  
    return dayOfMonth;  
}
```

```
/** The name */  
Private String name;
```

```
/** The version */  
Private String version;
```

```
/** The licenceName */  
Private String LicenceName;
```

이런 주석은 삭제해야 합니다

이력 및 저자를 기록하는 주석 SCM 도구를 사용해서 해결할 것

```
/*  
11-Oct-2001  
    작성자: 김민수  
    변경내용:  
    클래스를 다시 정리 하고 새로운 패키지 인  
    com.jrefinery.date로 옮김  
02-Nov-2001:  
    작성자: 홍길동  
    변경내용:  
    com.jrefinery.date로 옮김  
    NoitableDate 클래스 제거  
    getDescription() 메소드 추가  
*/
```

```
    | hFile = CreateFile(strName, GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_ALWAYS /*CREATE_ALWAYS*/,  
    //youyou_2009_03_18_OPEN_ALWAYS FILE_ATTRIBUTE_NORMAL, NULL);
```


이런 주석은 삭제해야 합니다

주석 처리한 코드

```
CString strDir = _T("");
CString strName = _T("");
strDir = (LPTSTR)szDir;
strName = (LPTSTR)szName;

/*
if(strDir.GetLength() > PATHSIZE*2 || strName.GetLength() > PATHSIZE*2)
{
    TRACE(_T("[NetFILE_MAKEDIR]Path and Name is Too Long(CString)\n"));
    LogMessage(si->m_ChannelInfo->strId, "[NetFILE_MAKEDIR]Path and Name is Too Long(CString)\n");
    return;
}*/

else if(pRemove->cOListType == LIST_SHARE)
{
    /* // youyou_NoCheck_StorageSize_AtShare
    if(objFcFile.FileDir_IsExist(strFullTargetPath))
    {
        if(strFileType == _T("FLDR"))
        {
            ullSize = objFcFile.GetUsedDisk(strFullTargetPath); //[Byte]
        }
        else
        {
            objFcFile.GetFileLength(strFullTargetPath, &ullSize);
        }
        PlusShareUsedQuota(si, strShareIdentity, ullSize);
    }
    else
    {
    }
    }*/
}
// youyou_ShareQuota end
```

오류 코드보다 예외를 사용하라

오류를 처리하기 위한 코드가 로직에 포함되면 가독성이 떨어진다.

Try ~ Catch문을 이용한 예외처리를 통해 로직과 오류처리를 분리한다.

```
if (deletePage(page) == E_OK) {  
    if(registry.deleteReference(page.name) == E_OK){  
        logger.log("page deleted");  
    }else{  
        logger.log("deleteReference from registry is failed");  
    }else{  
        logger.log("page deleted failed");  
    }  
}
```

```
try{  
    deletePage(page);  
    registry.deleteReference(page.name);  
}  
catch( Exception e){  
    logger.log(e.getMessage());  
}
```

1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?
4. Do you have a bug database?
5. Do you fix bugs before writing new code?
6. Do you have an up-to-date schedule?
7. Do you have a spec?
8. Do programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?