SysClinic 取扱説明書

更新日:2024/08/5

目次

本ソフトウェアについて **4**

動作環境 **4**

推奨ＰＣスペック **4**

画面説明 **5**

分析前の注意事項 **6**

分析方法 **6**

SysClinic リポジトリの新規作成 6

プロジェクトのインポート *7*

手動でプロジェクトを追加 *8*

Java 8

C# 9

プロジェクトの設定画面について *10*

分析 11

機能 **12**

リポジトリツリー 12

マトリクス 13

クラス継承マトリクス *13*

クラス参照マトリクス *14*

関数マトリクス *15*

変数マトリクス *16*

CRUD マトリクス *17*

SQL クエリの取得について 18

関連図 19

クラス継承関連図 *19*

クラス参照関連図 *20*

関数関連図 *21*

変数関連図 *22*

シーケンス図 23

フローチャート 24

一覧 25

クラス一覧 *25*
クラス一覧の項目について 26
関数一覧 *27*
関数一覧の項目について 28
変数一覧 *29*
検索 30
エクスポート ≪製品版のみ≫ 31
Excel へ出力 ≪製品版のみ≫ *31*
CSV へ出力 ≪製品版のみ≫ *31*

本ソフトウェアについて

SysClinic はソースコードを元にマトリクス、関連図、シーケンス図、フローチャートなどからソースコード構造を視覚的に参照可能にします。そしてクラスや関数のメトリクスを計測結果からソースコード構造の傾向を見ることもできます。またこれらの機能はすべて Excel もしくは CSV へ出力することができ、ドキュメントの部品として使用いただくこともできます。

動作環境

・Windows 10
・常時ネット接続環境が必要(※ライセンスの確認を行うため)

推奨ＰＣスペック

・ＣＰＵ　・・・・・　Core i5 以上(８世代以降)/ Ryzen 2300X 以上
・ＲＡＭ　・・・・・　16GB

画面説明

以下の画像は、SysClinic で java のソースコードを分析後の画面です。

❶メインメニュー

SysClinic リポジトリの作成や読み込みなどを行います。

❷リポジトリツリー

分析結果がワークスペースを根とし、その下にプロジェクト、ファイル、パッケージ(もしくは名前空間)、クラス、メソッド、フィールドがツリー上に表示されます。

❸ドキュメントエリア

リポジトリツリーから表示したマトリクス、関連図、一覧、シーケンス図、フローチャート、ソースコードが表示されます。

❹検索結果・タスクリスト・エラーリスト

検索結果を表示する検索結果ウィンドウ、
分析中の情報やエクスポート中の情報を表示するタスクリスト、
分析中のエラー情報を表示するエラーリストがあります。

分析前の注意事項

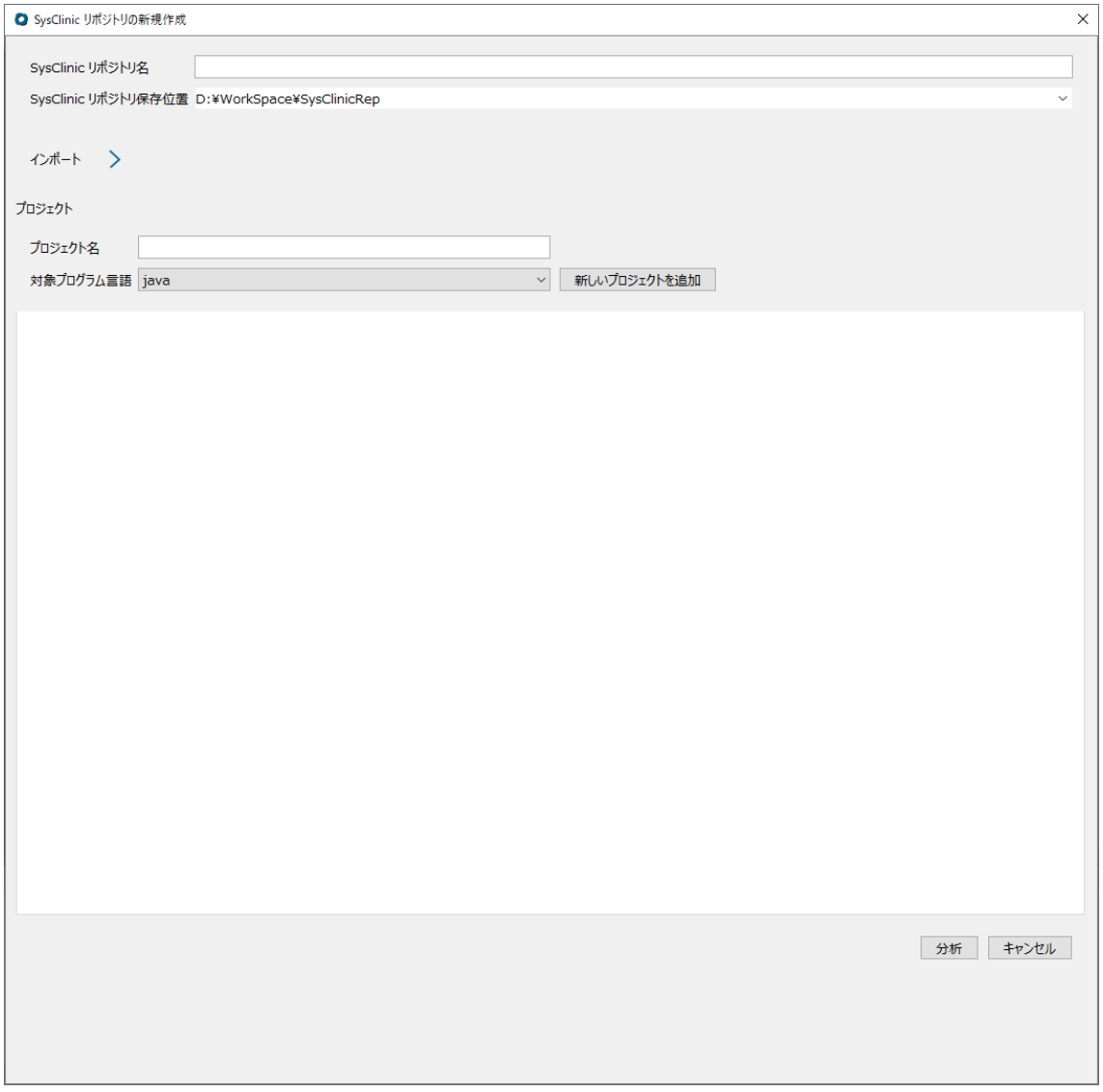
分析を行う際、分析対象のソースコードを開発しているＰＣで分析を行うことをお勧めします。開発ＰＣでの分析が難しい場合は、SysClinic をインストールするＰＣに開発ＰＣの環境を設定してください。

分析方法

分析するにはまず SysClinic リポジトリの作成を行います。

SysClinic リポジトリの新規作成

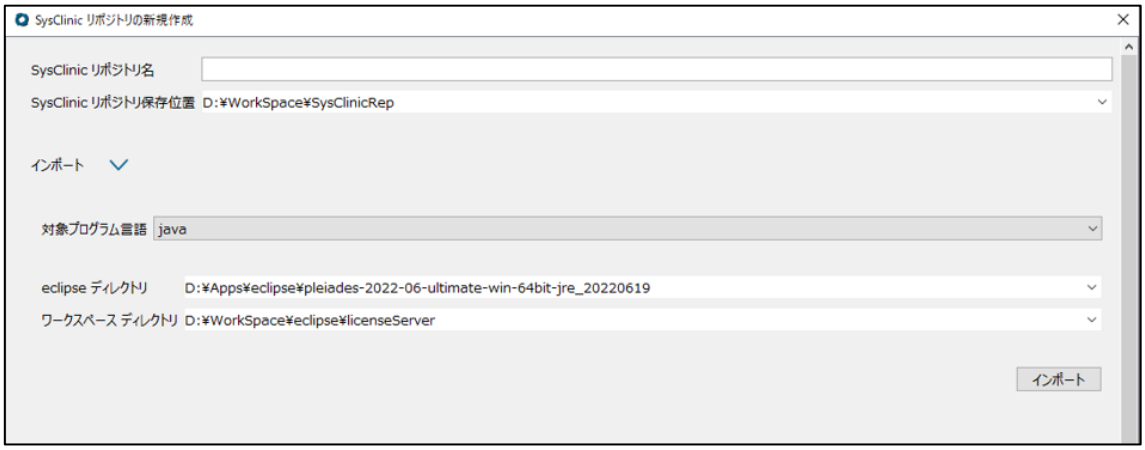
メインメニューの「ファイル」→「新規作成」より、SysClinic リポジトリ新規作成ダイアログを表示します。まずリポジトリ名を記入してください。



プロジェクトのインポート

インポート方法は以下を参照ください。

Java

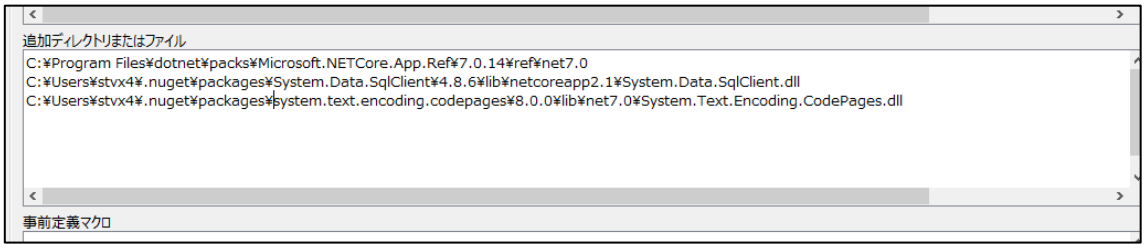


eclipse プロジェクトの maven(pom の継承がある場合は version 3.8 以上)
インポートを行うには eclipse ディレクトリパスと分析対象のワーク
スペースディレクトリのパスを設定し、インポートボタンを押します。

C#

Visual Studio 2022 以降のソリューションファイルを設定し、インポートボタンを押します。

※OracleClient など一部の DLL のパスが取れない場合があります。足りない DLL に関しましては手動で「追加のディレクトリまたはファイル」欄に DLL のフルパスを設定して下さい。



手動でプロジェクトを追加

手動でプロジェクトを追加する場合は、プロジェクト欄のプロジェクト名の記入と
対象プログラム言語の選択を行い新しいプロジェクトの追加を行います。
プログラム言語により、以下の設定時注意事項があります。

Java

「追加のディレクトリまたはファイル」欄に、ソースコードクラスパスと、ご使用の Java ランタイムの jar ファイルを記入する必要があります。

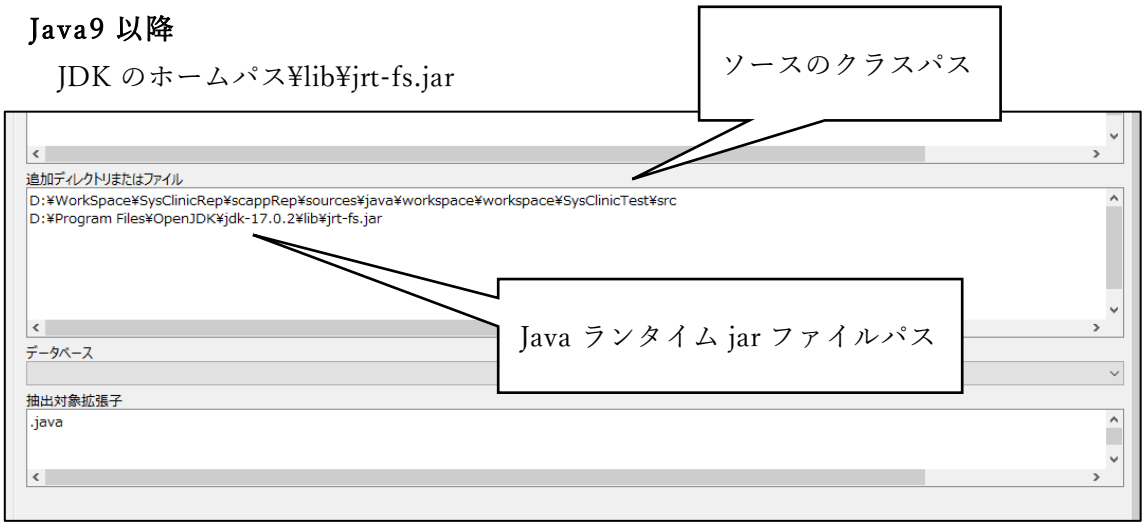
Java ランタイム対象となる jar ファイルは以下です。

Java8 以前

JDK のホームパス¥lib¥rt.jar
JDK のホームパス¥lib¥charsets.jar
JDK のホームパス¥lib¥jce.jar
JDK のホームパス¥lib¥jsse.jar
JDK のホームパス¥lib¥resources.jar

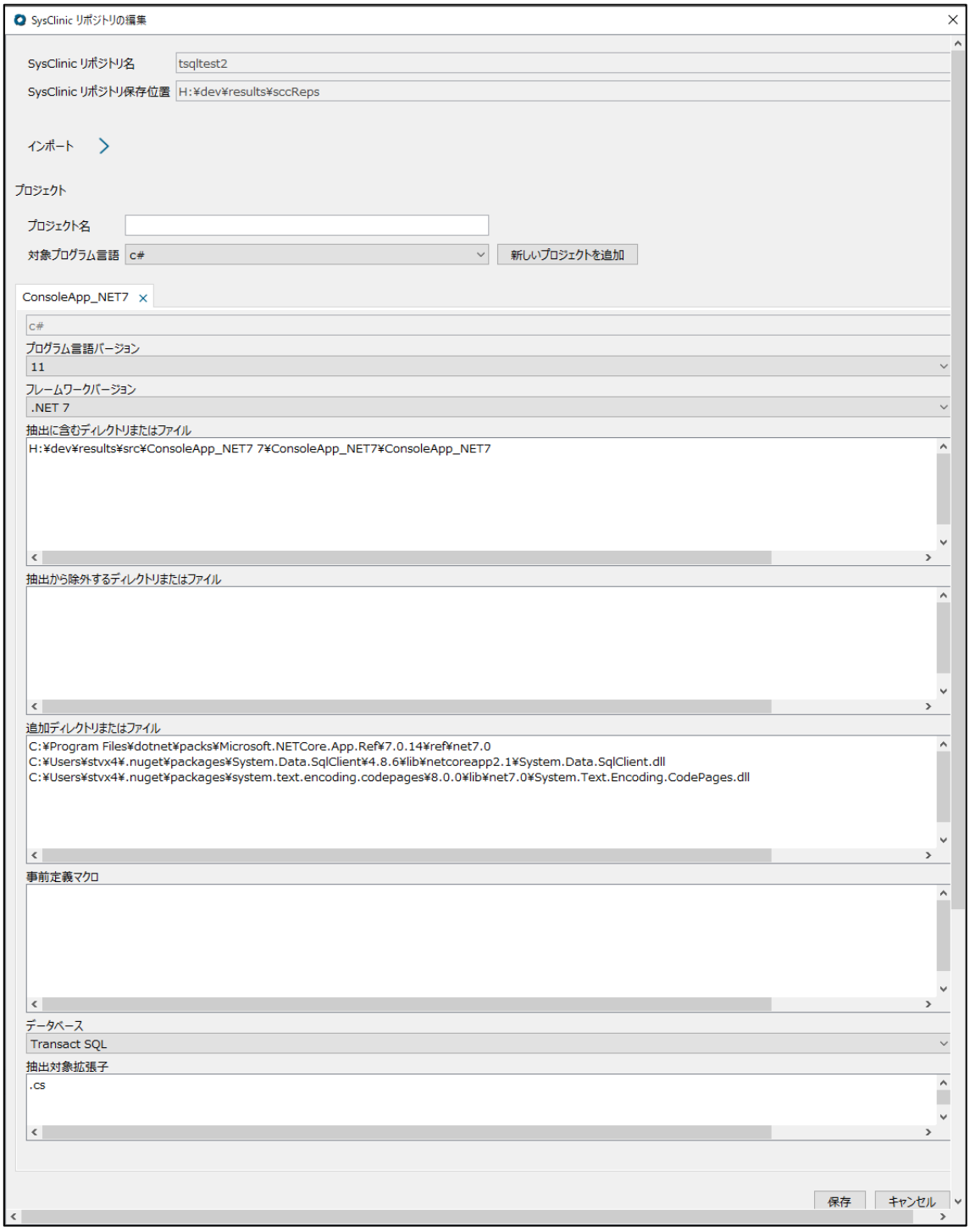
Java9 以降

JDK のホームパス¥lib¥jrt-fs.jar

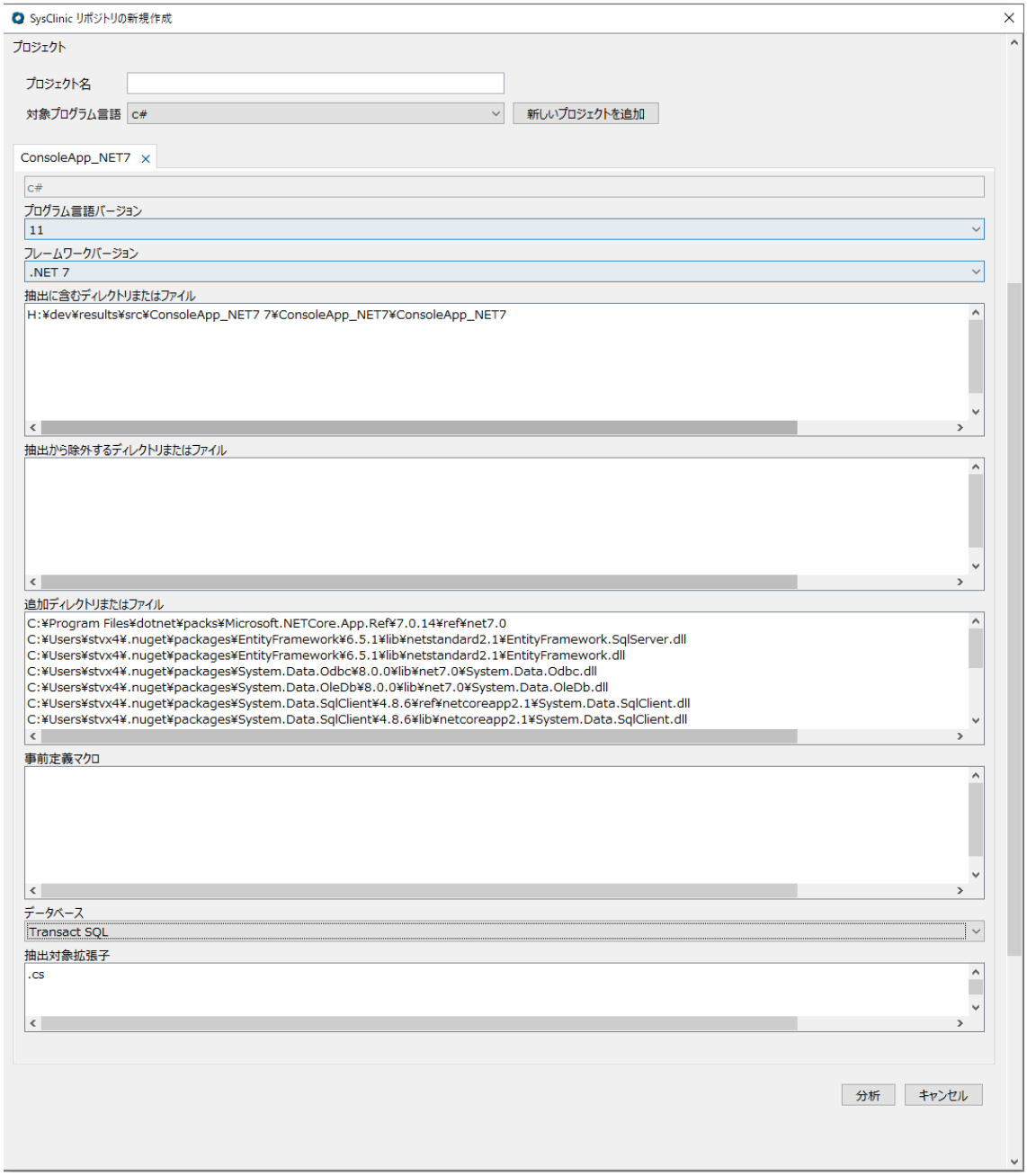


C#

まず、ソリューションのインポートを行ったあと、足りない情報があれば編集を行ってください。



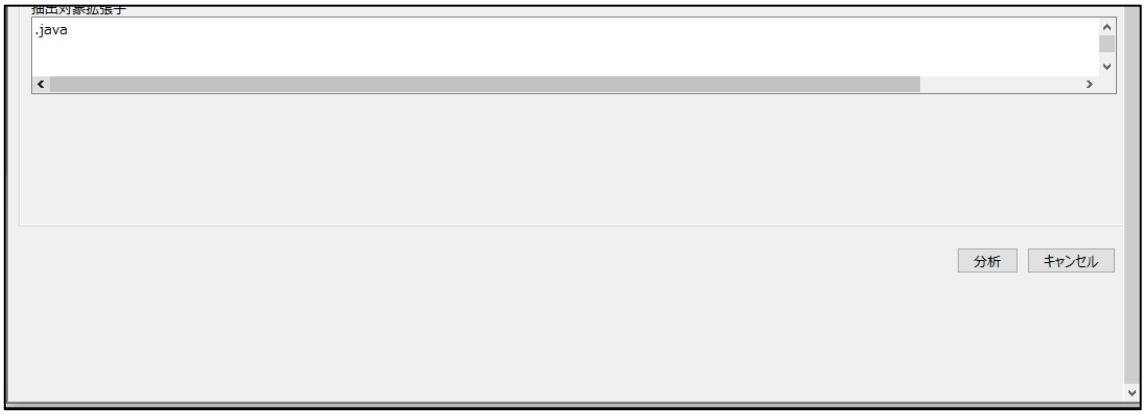
プロジェクトの設定画面について

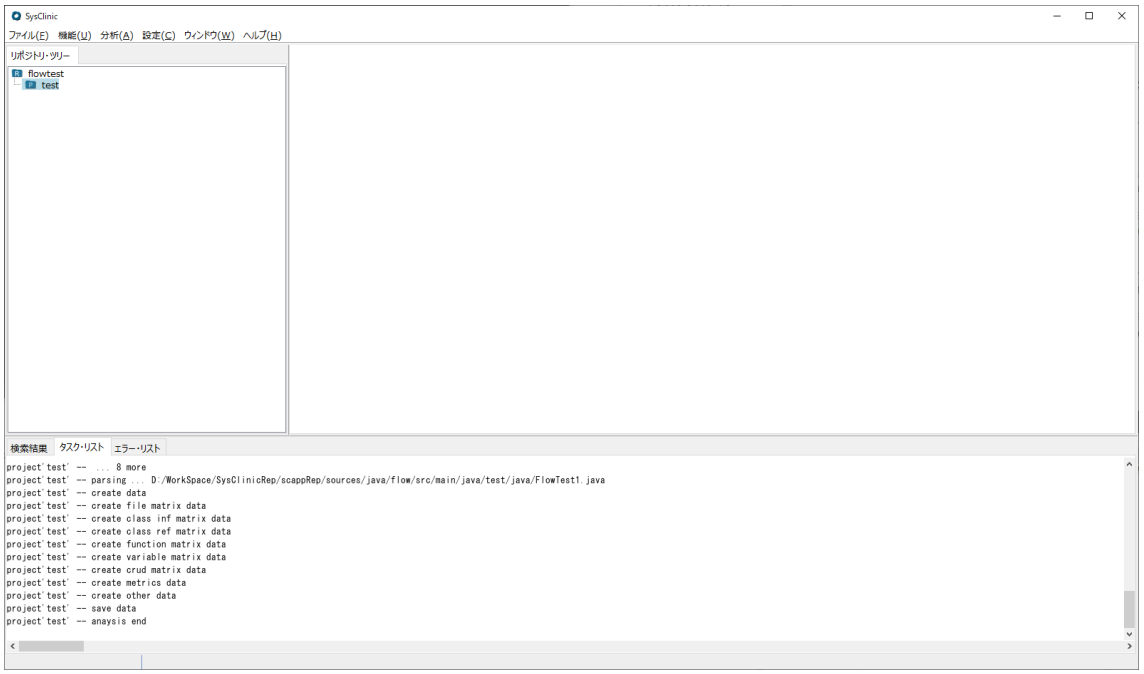


分析

プロジェクトの設定が完了しましたら、SysClinic リポジトリ新規作成ダイアログの一番下にある、分析ボタンを押します。

しばらくすると分析が終わり、分析結果が表示されます。



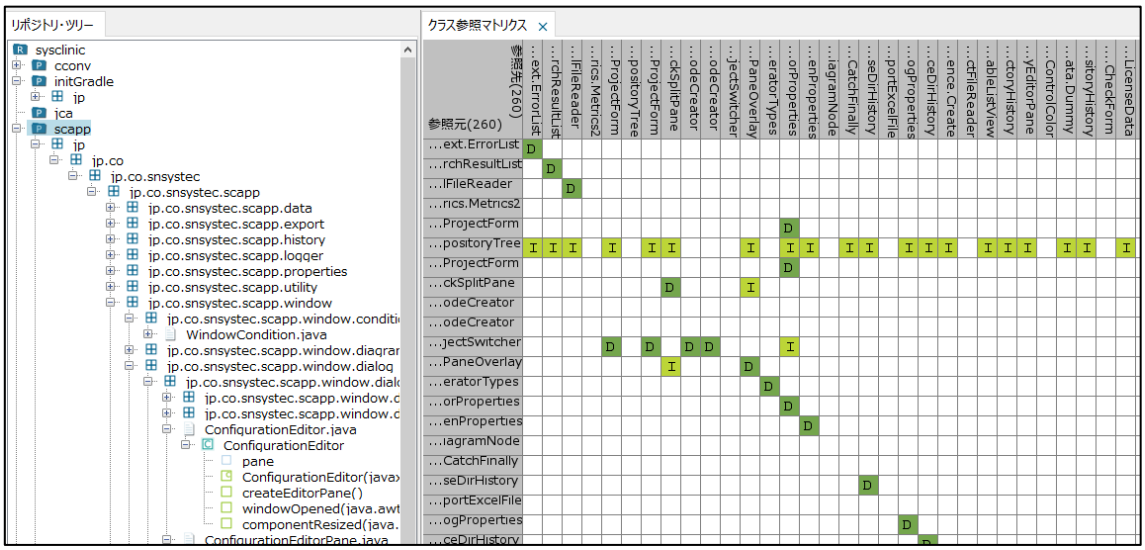


機能

リポジトリツリー

リポジトリツリーは、ワークスペースのルートから始まり、その下にプロジェクト、プロジェクトの下にパッケージ名、クラス名などが続きます。選択したノードの右クリック・メニューからマトリクスや一覧などを表示することにより、絞り込んで表示することができます。

以下は、プロジェクト scapp からクラス参照マトリクスを表示した画像と
パッケージ jp.co.snsystec.scapp.export からクラス参照マトリクスを表示した画像に
なります。下のほうの画像がより絞り込まれて表示されていることがわかります。



マトリクス

左と上にヘッダがあり、ヘッダ同士の相関関係を表した表になります。
相関関係がある場合は、関係の意味を表すアルファベットが表示されます。

クラス継承マトリクス

リポジトリツリーのノードもしくは、メインメニューの機能より表示します。

ヘッダについて

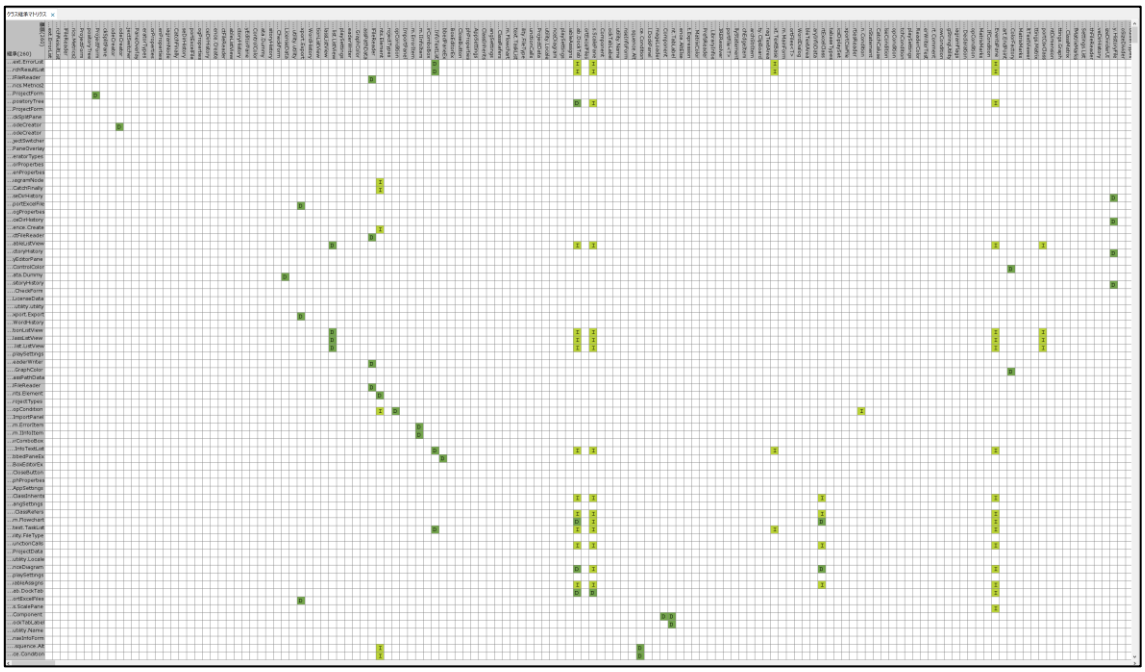
上のヘッダは基底クラスが並びます。

左のヘッダは継承クラスが並びます。

マークについて

D・・・直接(direct)の継承関係を表します。

I ・・・関節(indirect)の継承関係を表します。



クラス参照マトリクス

リポジトリツリーのノードもしくは、メインメニューの機能より表示します。

ヘッダについて

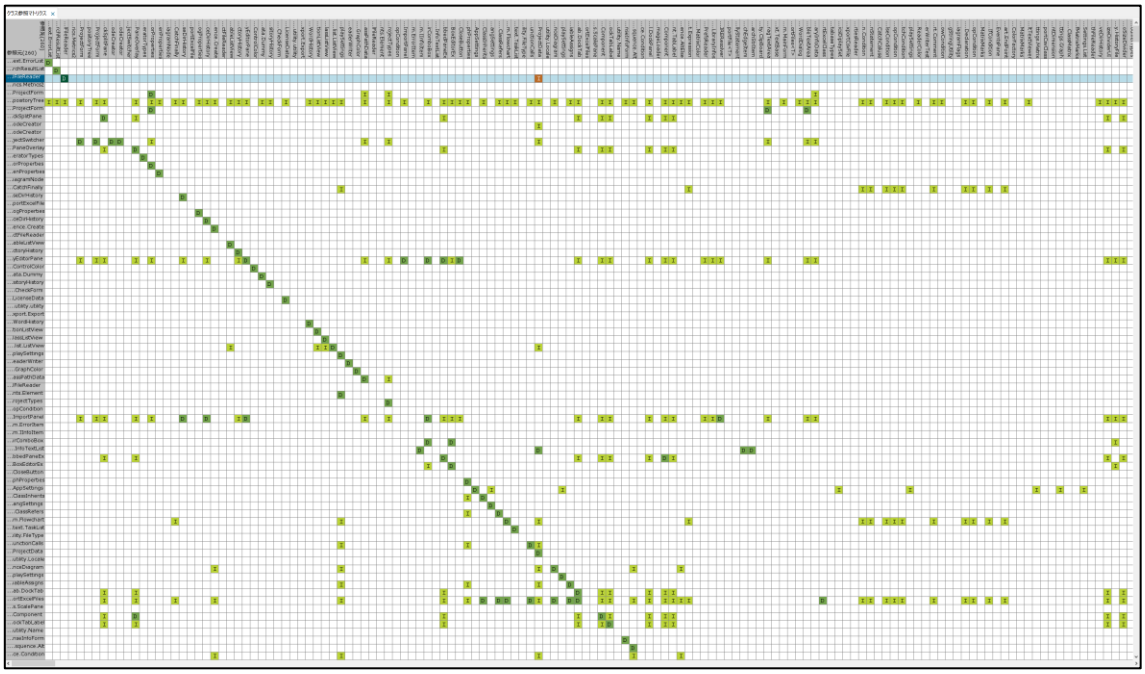
上のヘッダは参照先クラスが並びます。

左のヘッダは参照元クラスが並びます。

マークについて

D・・・直接(direct)の参照関係を表します。

Ｉ・・・関節(indirect)の参照関係を表します。



関数マトリクス

リポジトリツリーのノードもしくは、メインメニューの機能より表示します。

ヘッダについて

上のヘッダは参照先関数が並びます。

左のヘッダは参照元関数が並びます。

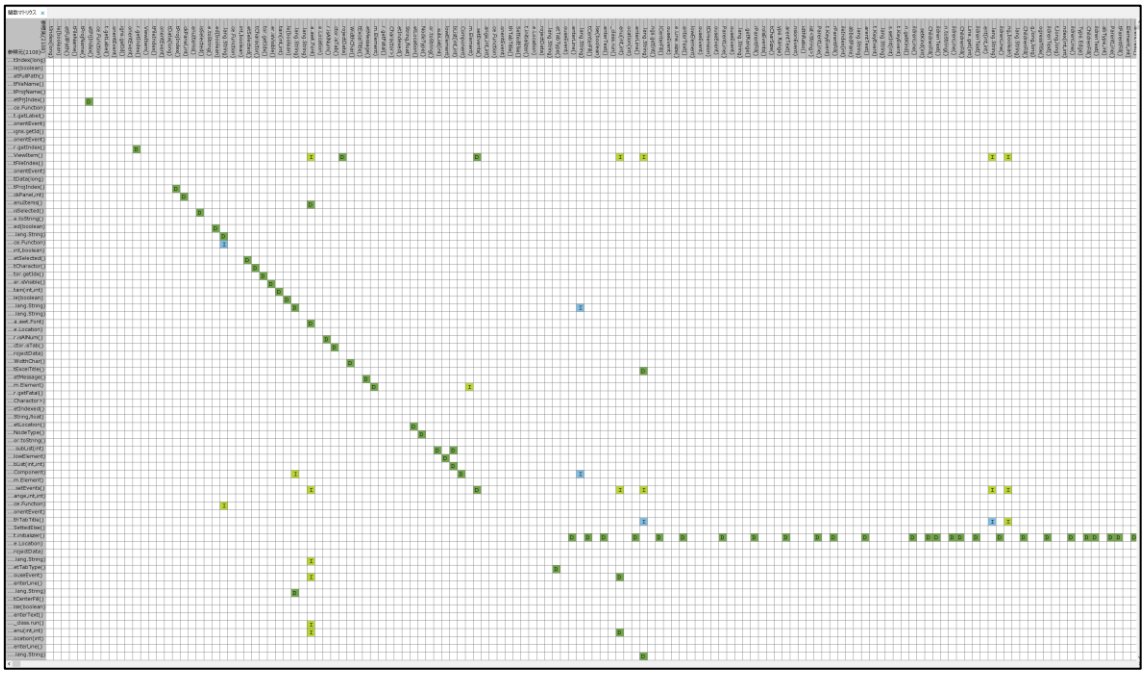
マークについて

緑D　・・・直接(direct)のコール関係を表します。

黄緑Ｉ・・・関節(indirect)のコール関係を表します。

青Ｄ　・・・override 関数を介した直接(direct)のコール関係を表します。

空色Ｉ・・・override 関数を介した関節(indirect)のコール関係を表します。



変数マトリクス

リポジトリツリーのノードもしくは、メインメニューの機能より表示します。

ヘッダについて

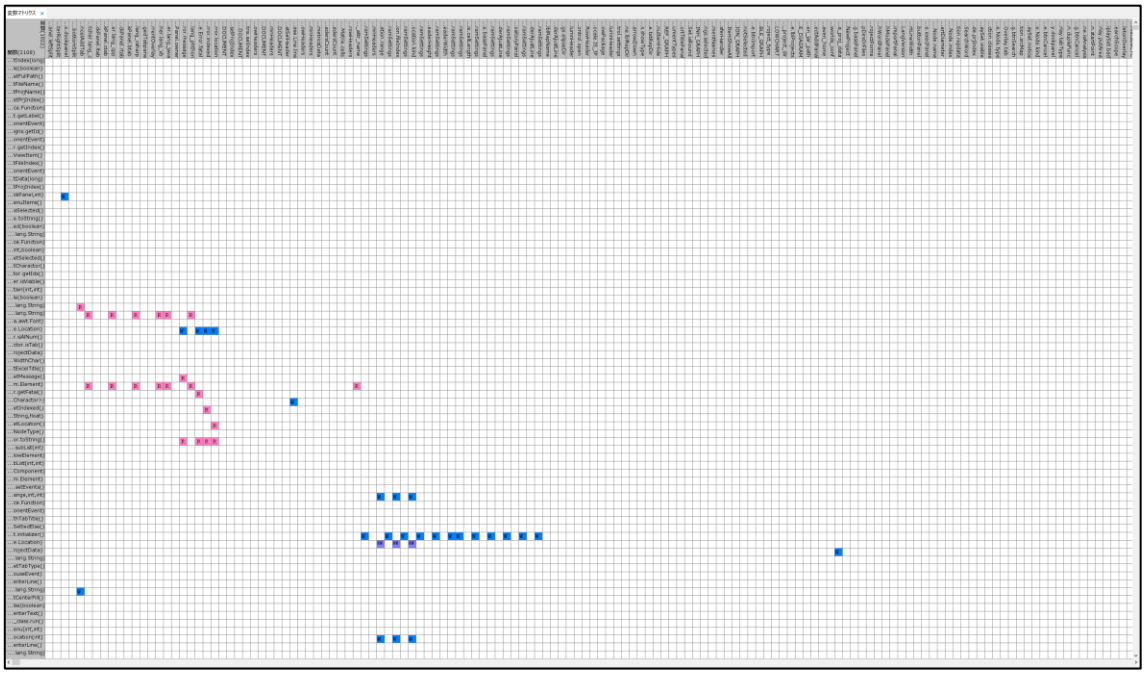
上のヘッダはグローバル変数もしくはフィールド、メンバ変数が並びます。

左のヘッダは関数が並びます。

マークについて

R・・・参照を表します。

W・・・書き込み表します。



CRUD マトリクス

リポジトリツリーのノードもしくは、メインメニューの機能より表示します。

ヘッダについて

上のヘッダは DB カラムが並びます。

左のヘッダは関数が並びます。

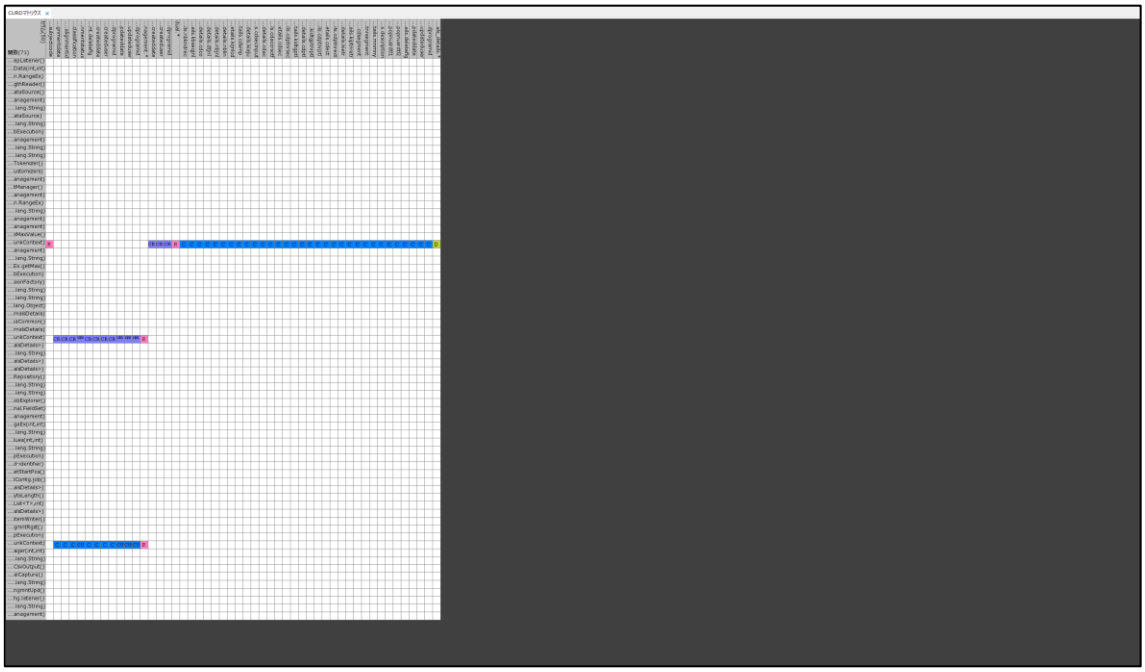
マークについて

C・・・Insert 文を表します。

R・・・Select 文を表します。

U・・・Update 文を表します。

D・・・Delete 文を表します。



SQL クエリの取得について

CRUD 図を作成する際、ソースコードにハードコーディングされているクエリから CRUD 図を作成しています。

Java の mybatis は XML よりクエリを取得し CRUD 図を作成します。

Java

Java での SQL 取得対象は以下となります。

・mybatis

C#

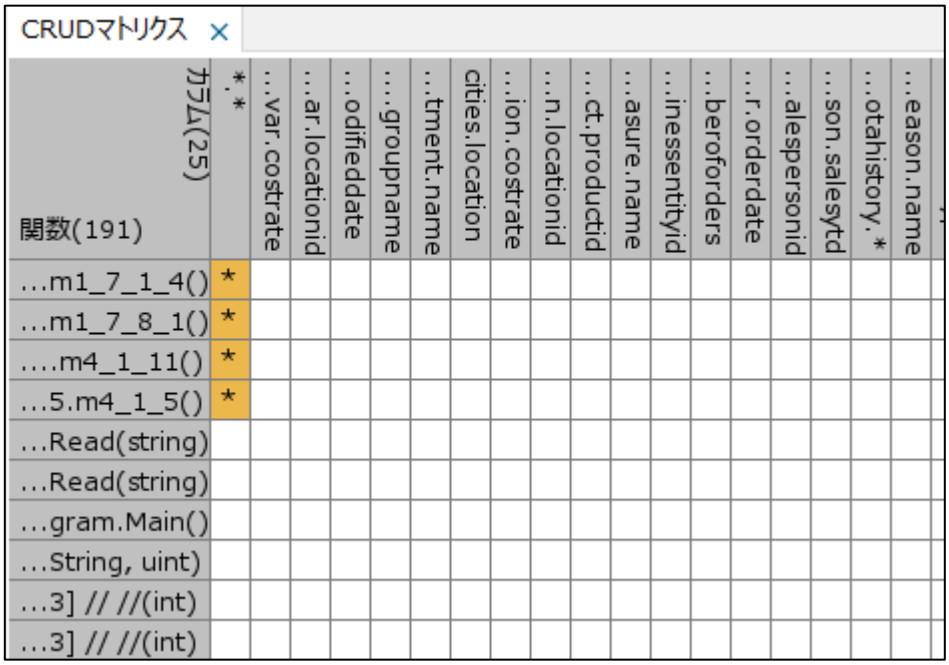
C#での SQL 取得対象は以下となります。

・SqlDataCommand

・SqlDataAdapter

ただし SqlDataAdapter で自動生成されたクエリに関しては詳しい結果は出ません。

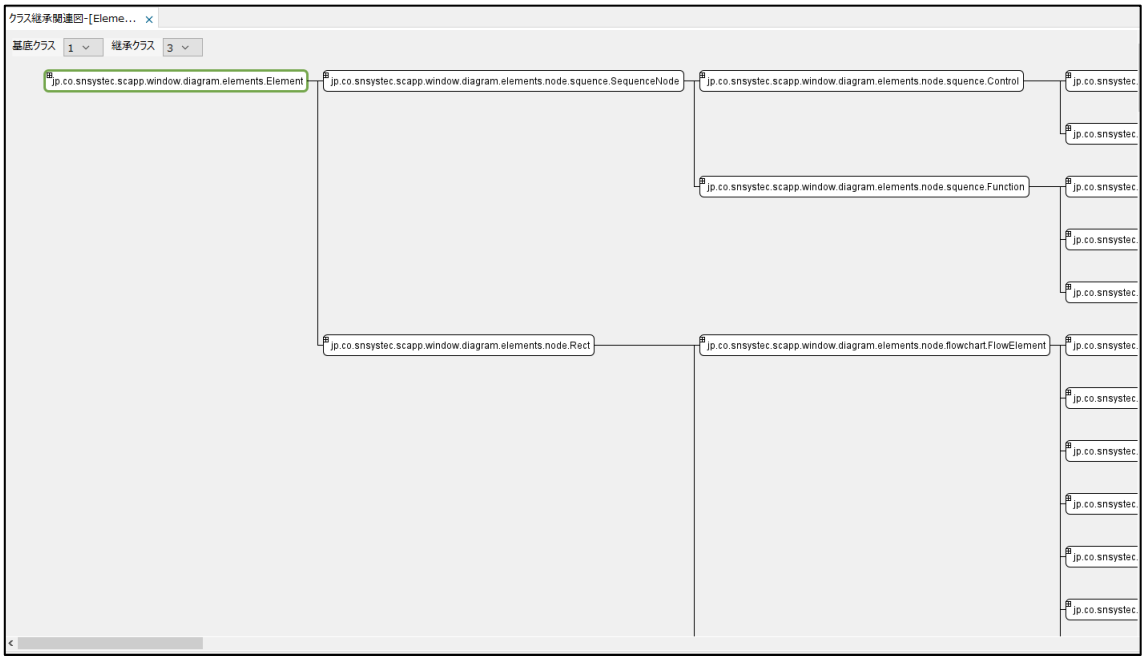
この場合、*マークが表示されます。



関連図

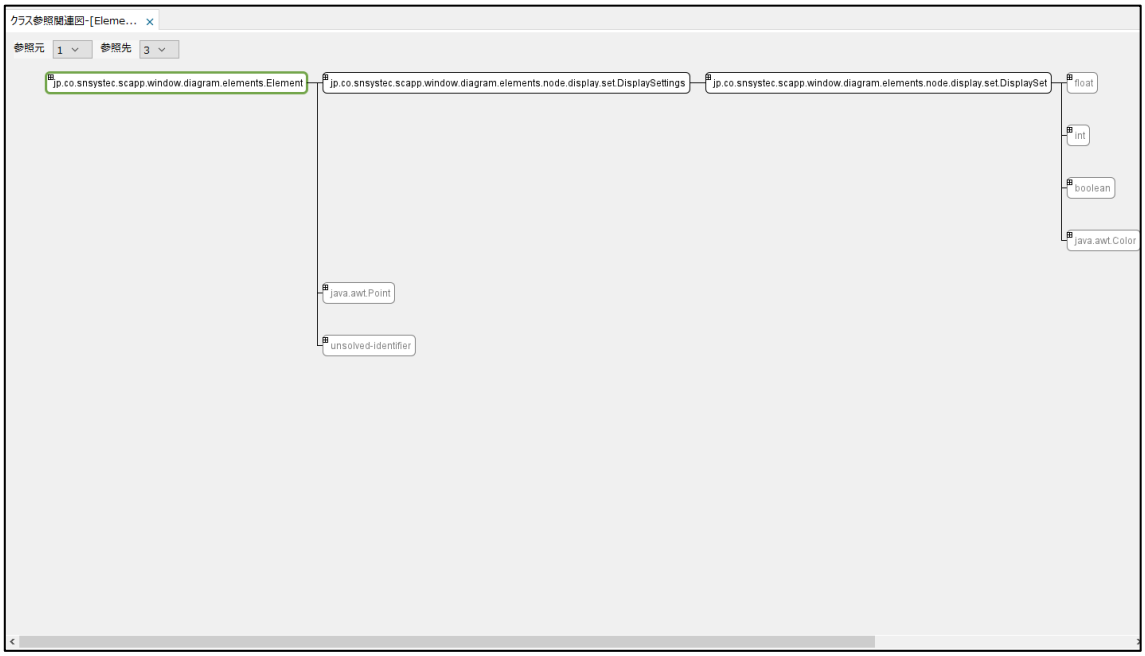
クラス継承関連図

リポジトリツリーのクラスノード、クラスマトリクス、クラス一覧より表示します。



クラス参照関連図

リポジトリツリーのクラスノード、クラスマトリクス、クラス一覧より表示します。



関数関連図

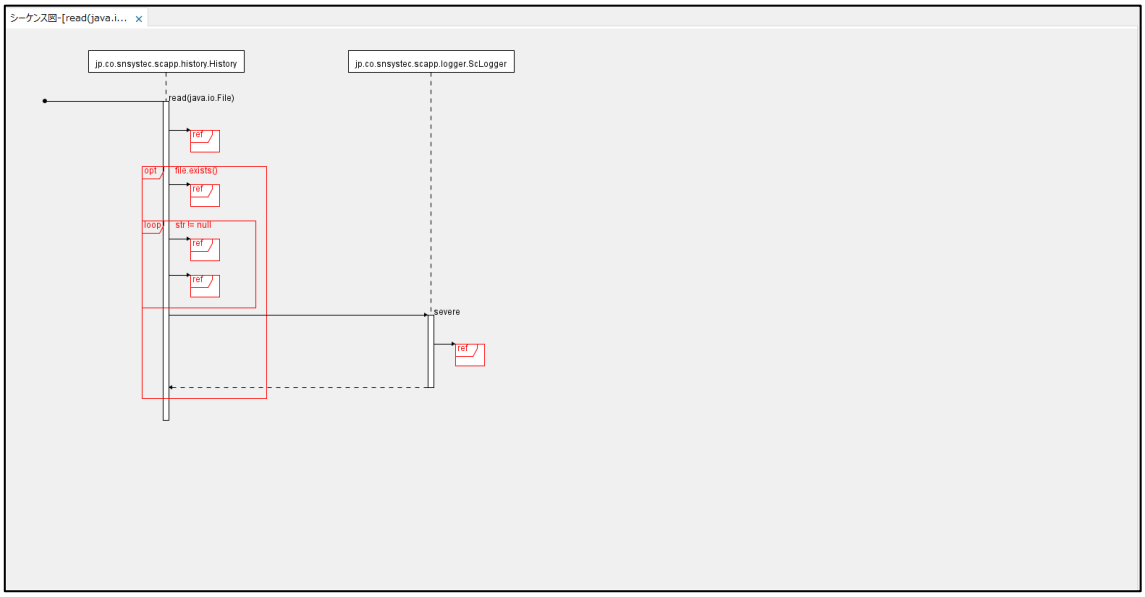
リポジトリツリーの関数ノード、関数マトリクス、関数一覧より表示します。

変数関連図

リポジトリツリーの変数ノード、変数マトリクス、変数一覧より表示します。
選択した変数と代入関係がある場合に関連が表示されます。

シーケンス図

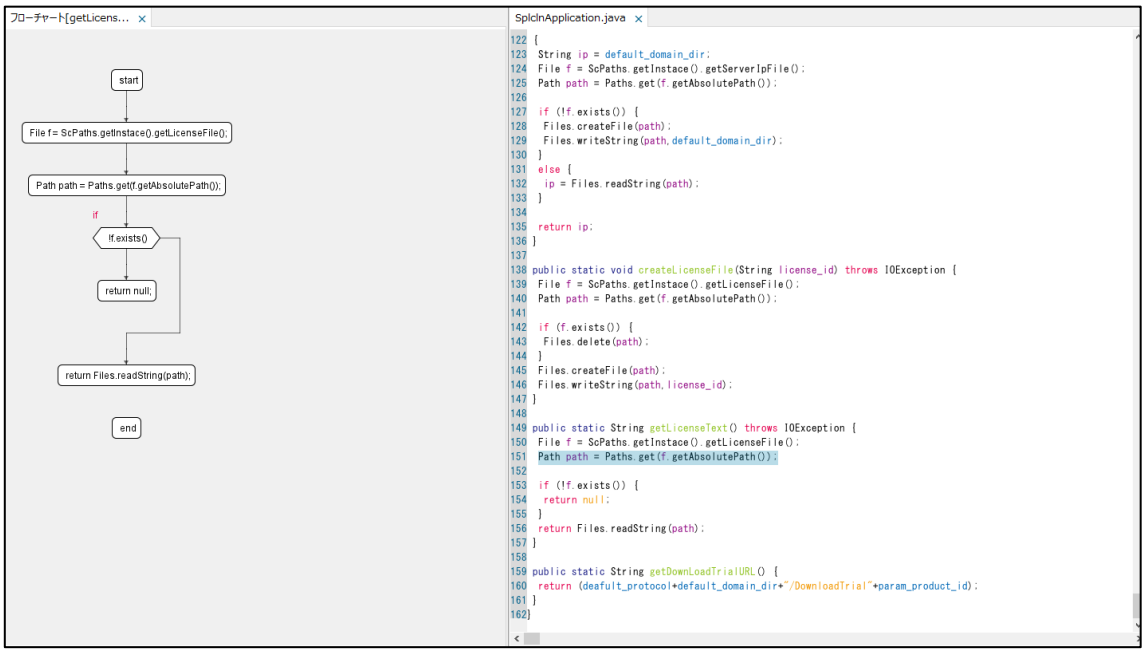
リポジトリツリーの関数ノード、関数マトリクス、関数一覧より表示します。
選択した関数が左側に表示され、時系列順に処理や関係のある他の関数などが表示されます。



フローチャート

リポジトリツリーの関数ノード、関数マトリクス、関数一覧より表示します。

フローチャートのノードをクリックすると、ソースコードが表示されます。

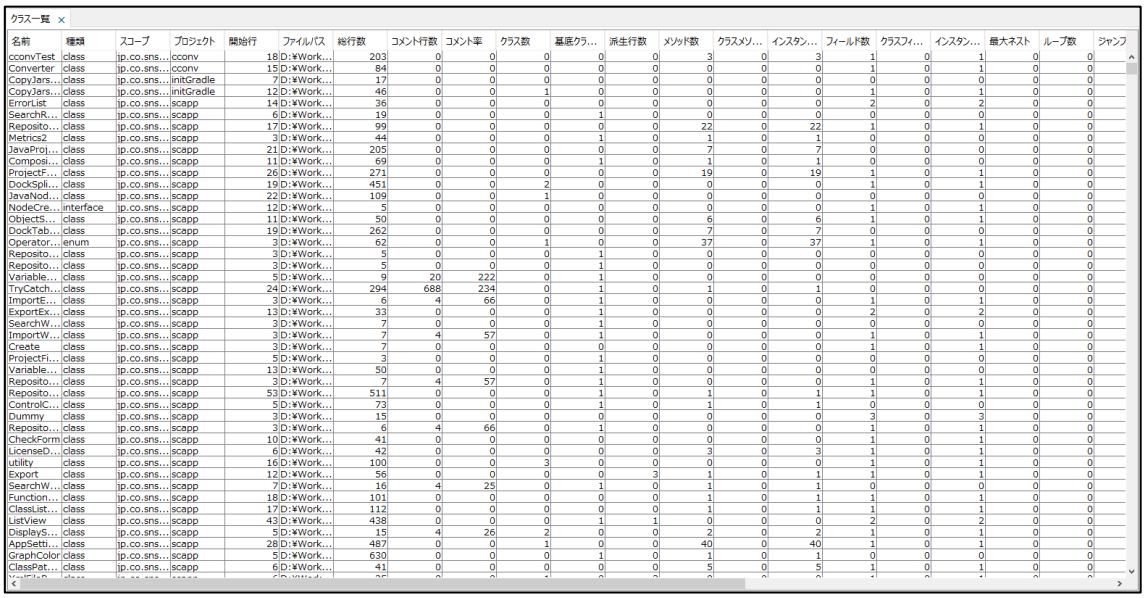


一覧

リポジトリツリーのノードもしくは、メインメニューの機能より表示します。

クラス一覧

リポジトリツリーのクラスノード、メインメニューの機能、クラスマトリクス、クラス一覧より表示します。

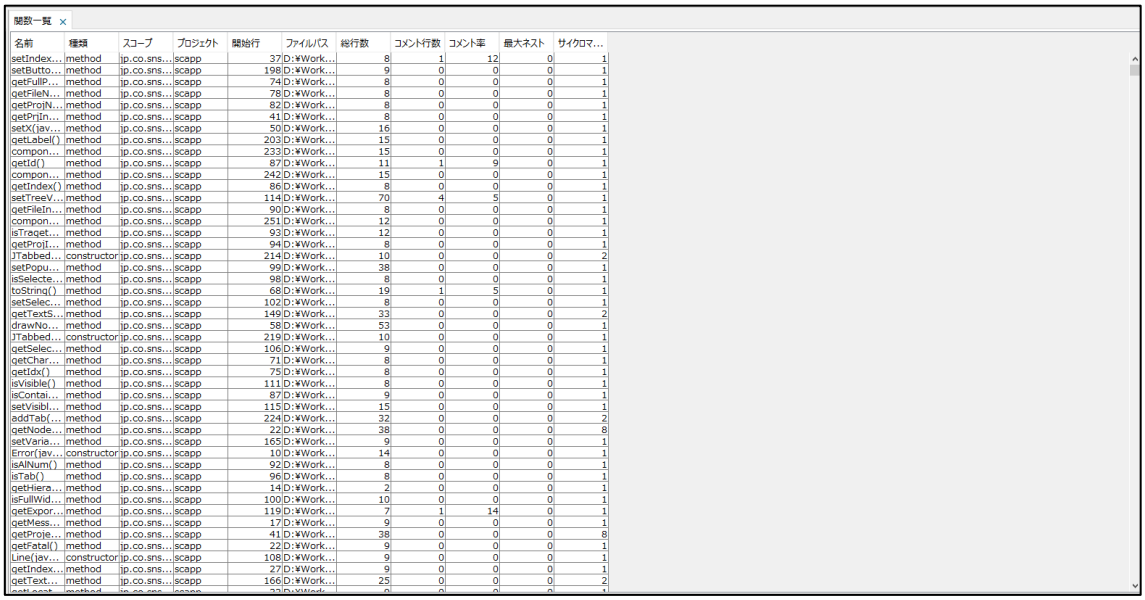


クラス一覧の項目について

総行数　　・・・　クラス定義の行数
コメント行数・・・　クラス定義内にあるコメント行数
コメント率　・・・　コメント行数／総行数×１００
クラス数　　・・・　クラス内定義されているクラス数
基本クラス数
派生クラス数
メソッド数　・・・　クラスメソッド数＋インスタンスメソッド数
クラスメソッド数・・・　static メソッドの数
インスタンスメソッド数・・・　static でないメソッドの数
フィールド数・・・　クラスフィールド数＋インスタンスフィールド数
クラスフィールド数・・・　static フィールドの数
インスタンスフィールド数・・・　static でないフィールド数
最大ネスト・・・　制御文のネストの最大値
ループ数・・・　ループ文の総数
ジャンプ数・・・　ジャンプ文の総数
goto・・・　goto 分の数
CK__WMC・・・　CK メトリクス Weighted Methods Per Class(WMC)、クラス内のメソッドのサイクロマティック複雑度の総数
CK__DIT・・・　CK メトリクス Depth of Inheritance Tree(DIT)、基底クラスまでの継承の深さ
CK__NOC・・・　CK メトリクス Number of Children(NOC)、直接派生しているサブクラスの数
CK__CBO・・・　CK メトリクス Coupling between Object Classes(CBO)結合しているクラスの数
CK__RFC・・・　CK メトリクス Response for a Class(RFC)、メソッドから呼び出されるメソッド数
CK__LCOM・・・　CK メトリクス Lack of Cohesion of Methods(LCOM)、クラス内のメソッドの凝集性
CC(平均)　・・・　クラス内のメソッドのサイクロマティック複雑度の平均値
CC(最大)　・・・　クラス内のメソッドのサイクロマティック複雑度の最大値
CC(総数)　・・・　クラス内のメソッドのサイクロマティック複雑度の総数

関数一覧

リポジトリツリーの関数ノード、メインメニューの機能、関数マトリクス、関数一覧より表示します。


関数一覧

名前	種類	スコープ	プロジェクト	開始行	ファイルパス	総行数	コメント行数	コメント率	最大ネスト	サイクロマ…
setIndex…	method	jp.co.sns…	scapp	37	D:¥Work…	8	1	12	0	1
setButto…	method	jp.co.sns…	scapp	198	D:¥Work…	9	0	0	0	1
getFullP…	method	jp.co.sns…	scapp	74	D:¥Work…	8	0	0	0	1
getFileN…	method	jp.co.sns…	scapp	78	D:¥Work…	8	0	0	0	1
getProjN…	method	jp.co.sns…	scapp	82	D:¥Work…	8	0	0	0	1
getPrjIn…	method	jp.co.sns…	scapp	41	D:¥Work…	8	0	0	0	1
setX(jav…	method	jp.co.sns…	scapp	50	D:¥Work…	16	0	0	0	1
getLabel()	method	jp.co.sns…	scapp	203	D:¥Work…	15	0	0	0	1
compon…	method	jp.co.sns…	scapp	233	D:¥Work…	15	0	0	0	1
getId()	method	jp.co.sns…	scapp	87	D:¥Work…	11	1	9	0	1
compon…	method	jp.co.sns…	scapp	242	D:¥Work…	15	0	0	0	1
getIndex()	method	jp.co.sns…	scapp	86	D:¥Work…	8	0	0	0	1
setTreeV…	method	jp.co.sns…	scapp	114	D:¥Work…	70	4	5	0	1
getFileIn…	method	jp.co.sns…	scapp	90	D:¥Work…	8	0	0	0	1
compon…	method	jp.co.sns…	scapp	251	D:¥Work…	12	0	0	0	1
isTraget…	method	jp.co.sns…	scapp	93	D:¥Work…	12	0	0	0	1
getProjI…	method	jp.co.sns…	scapp	94	D:¥Work…	8	0	0	0	1
JTabbed…	constructor	jp.co.sns…	scapp	214	D:¥Work…	10	0	0	0	2
setPopu…	method	jp.co.sns…	scapp	99	D:¥Work…	38	0	0	0	1
isSelecte…	method	jp.co.sns…	scapp	98	D:¥Work…	8	0	0	0	1
toString()	method	jp.co.sns…	scapp	68	D:¥Work…	19	1	5	0	1
setSelec…	method	jp.co.sns…	scapp	102	D:¥Work…	8	0	0	0	1
getTextS…	method	jp.co.sns…	scapp	149	D:¥Work…	33	0	0	0	2
drawNo…	method	jp.co.sns…	scapp	58	D:¥Work…	53	0	0	0	1
JTabbed…	constructor	jp.co.sns…	scapp	219	D:¥Work…	10	0	0	0	1
getSelec…	method	jp.co.sns…	scapp	106	D:¥Work…	9	0	0	0	1
getChar…	method	jp.co.sns…	scapp	71	D:¥Work…	8	0	0	0	1
getIdx()	method	jp.co.sns…	scapp	75	D:¥Work…	8	0	0	0	1
isVisible()	method	jp.co.sns…	scapp	111	D:¥Work…	8	0	0	0	1
isContai…	method	jp.co.sns…	scapp	87	D:¥Work…	9	0	0	0	1
setVisibl…	method	jp.co.sns…	scapp	115	D:¥Work…	15	0	0	0	1
addTab(…	method	jp.co.sns…	scapp	224	D:¥Work…	32	0	0	0	2
getNode…	method	jp.co.sns…	scapp	22	D:¥Work…	38	0	0	0	8
setVaria…	method	jp.co.sns…	scapp	165	D:¥Work…	9	0	0	0	1
Error(jav…	constructor	jp.co.sns…	scapp	10	D:¥Work…	14	0	0	0	1
isAlNum()	method	jp.co.sns…	scapp	92	D:¥Work…	8	0	0	0	1
isTab()	method	jp.co.sns…	scapp	96	D:¥Work…	8	0	0	0	1
getHiera…	method	jp.co.sns…	scapp	14	D:¥Work…	2	0	0	0	1
isFullWid…	method	jp.co.sns…	scapp	100	D:¥Work…	10	0	0	0	1
getExpor…	method	jp.co.sns…	scapp	119	D:¥Work…	7	1	14	0	1
getMess…	method	jp.co.sns…	scapp	17	D:¥Work…	9	0	0	0	1
getProje…	method	jp.co.sns…	scapp	41	D:¥Work…	38	0	0	0	8
getFatal()	method	jp.co.sns…	scapp	22	D:¥Work…	9	0	0	0	1
Line(jav…	constructor	jp.co.sns…	scapp	108	D:¥Work…	9	0	0	0	1
getIndex…	method	jp.co.sns…	scapp	27	D:¥Work…	9	0	0	0	1
getText…	method	jp.co.sns…	scapp	166	D:¥Work…	25	0	0	0	2

関数一覧の項目について

総行数　　・・・　クラス定義の行数
コメント行数・・・　クラス定義内にあるコメント行数
コメント率　・・・　コメント行数／総行数×１００
最大ネスト・・・　制御文のネストの最大値
サイクロマティック複雑度・・・　メソッドのサイクロマティック複雑度数

サイクロマティック複雑度の計算方法

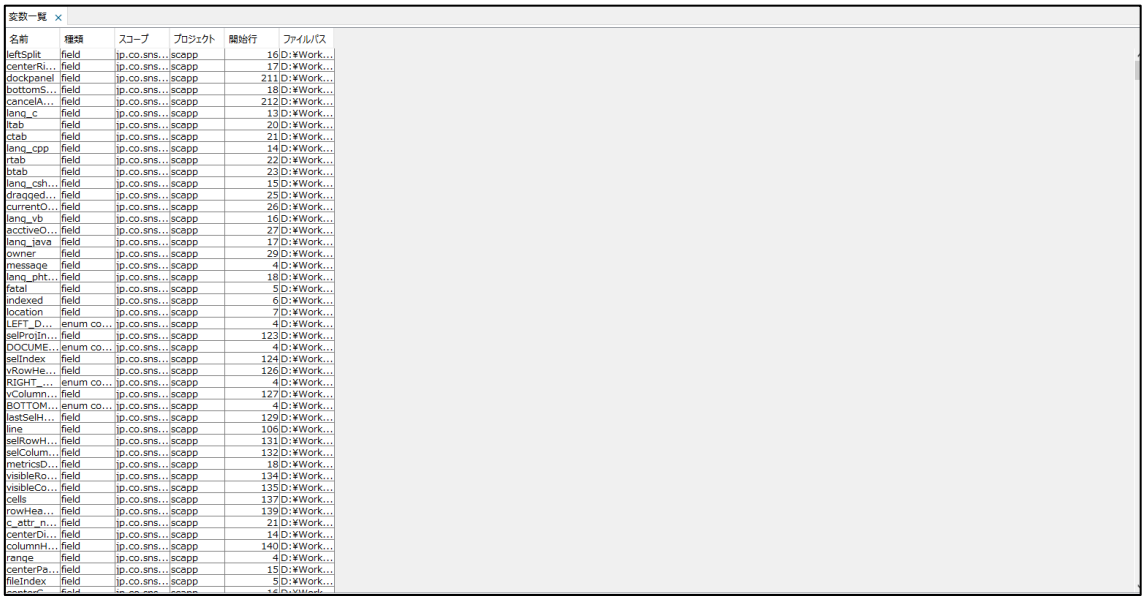
サイクロマティック複雑度＝条件文※の数＋１
※条件文(if, while, for , case など。else や default は対象外)

例
以下の関数 method についてのサイクロマティック複雑度は
1(if) + 1(case) + 1 = ３、となります。

```
int method(bool f, int i)
{
	if (f)
	{
		switch(i)
		{
		case 1:
			return 0;
		default:
			return -1;
		}
	}
	else
	{
		return 1;
	}
}
```

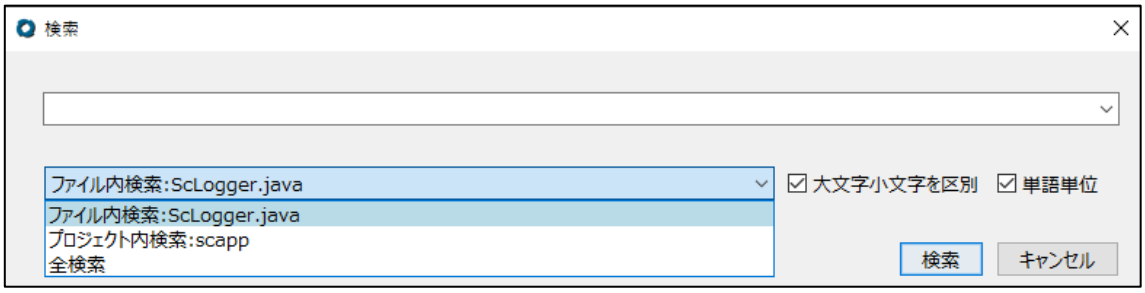
変数一覧

リポジトリツリーの変数ノード、メインメニューの機能、変数マトリクス、変数一覧より表示します。



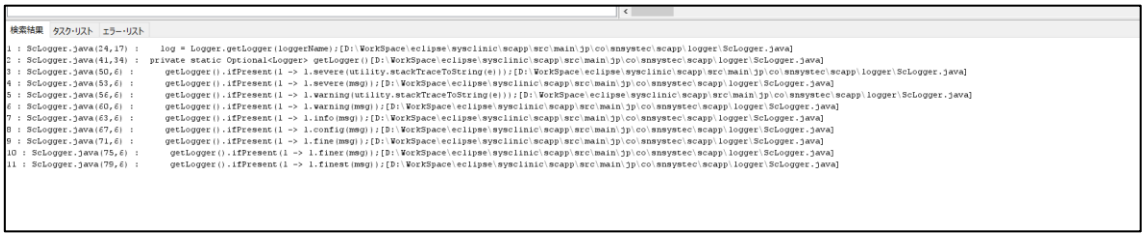
検索

Ctrl+F を押すことで検索ダイアログが開き検索を行うことができます。



検索結果は、画面下部に表示されます。

行をダブルクリックすることで該当ソースコードを表示することができます。



エクスポート ≪製品版のみ≫

メインメニューのエクスポートもしくはリポジトリツリーのノードの右クリックメニュー→エクスポートよりエクスポートを行うことができます。

Excel へ出力 ≪製品版のみ≫

マトリクス、関連図、シーケンス図、フローチャートを出力することができます。

CSV へ出力 ≪製品版のみ≫

一覧を出力することができます。

問い合わせ先
新日本システック株式会社
〒103－0004
東京都中央区東日本橋 1-2-6
Tell　: 03-5835-0671
e-mail: sns-solution@snsystec.co.jp