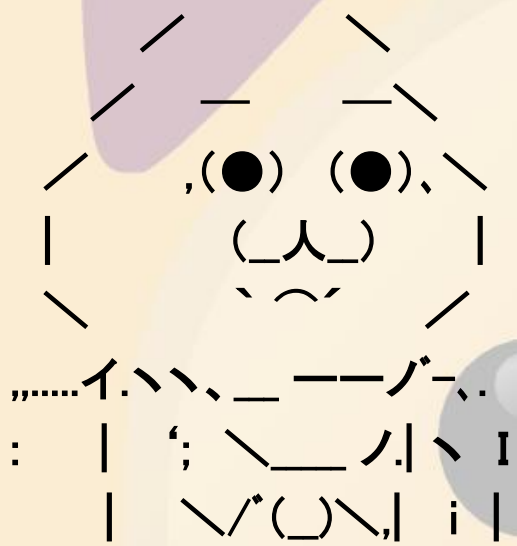
匠の伝承

マルチな時代の設計と開発

スピーカー自己紹介



ゆーちです。

ハンドル名です。

本名は、内山康広といいます。

47歳です。

おっさんです。_|￣|○

株式会社シーソフト代表取締役です。

現役のエンジニアです。プログラム書いてます。

メールソフト Becky! 用の BkReplyer という作品が微妙に有名らしす。
2ちゃんねらーではありません。

さいきん、「やるおがIT土方になったようです」を読みました。

株式会社シーソフト

独立系ソフトハウス

Seasoft Corporation＝海援隊から命名。
IT革命の寵児をめざし、1996年設立。

中小企業。

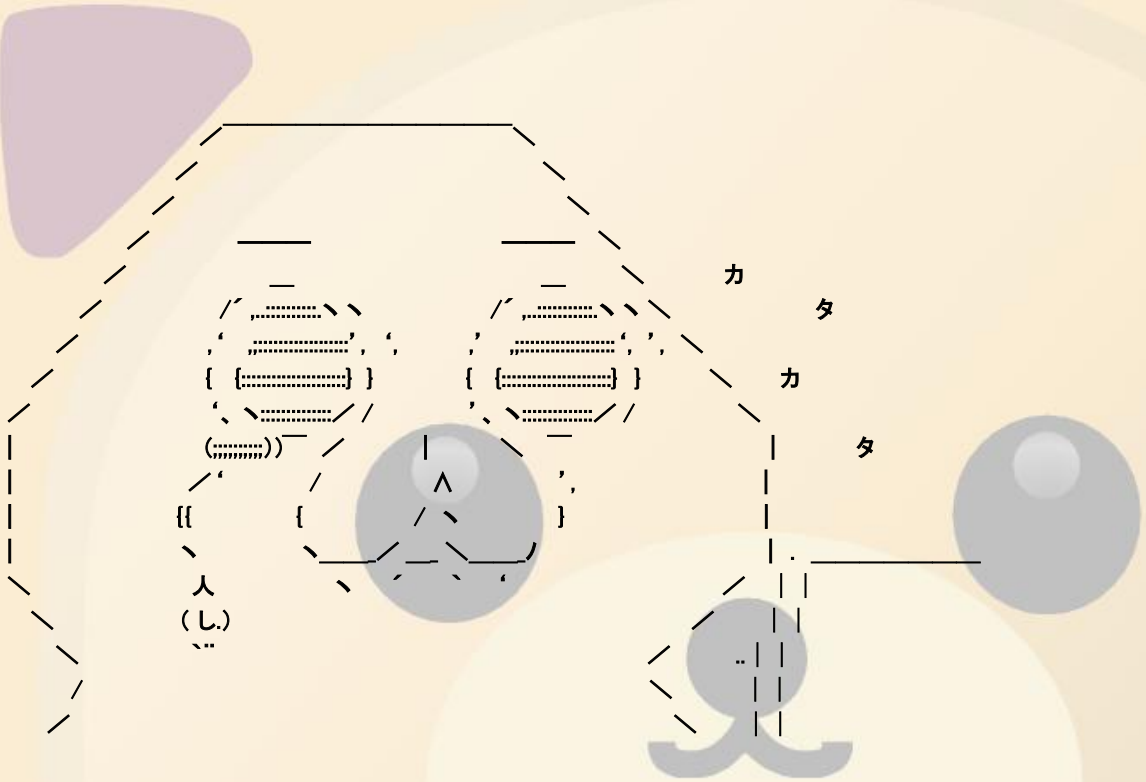
零細企業。

開発下請け、孫請け、ひ孫請け。

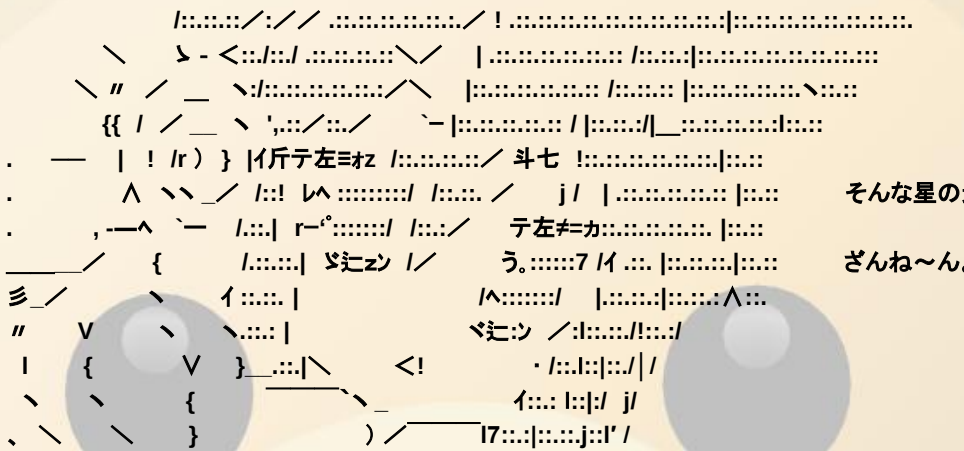
いつも潰れそうです。＿|￣|○

開発実績

・プリント基板設計用CAD
・金融機関Windowsベースプラットフォーム
・ホール用音響制御システム
・ホテル有料サービスカードシステム
・ゴミ処理場入出庫管理、課金システム
・河川道路のビデオ映像連動管理図面システム
・遠隔監視カメラ映像配信システム
・信用取引システム
・金融機関WEBベースフロー制御システム
・航空機用上空風力＆タービランスシステム
・溶融炉分析システム
・組み込み機器用ログデータ解析パッケージ
・テナントビル用デマンド管理
ほかにもいっぱい・・・



ほとんどが、デスマーチプロジェクト

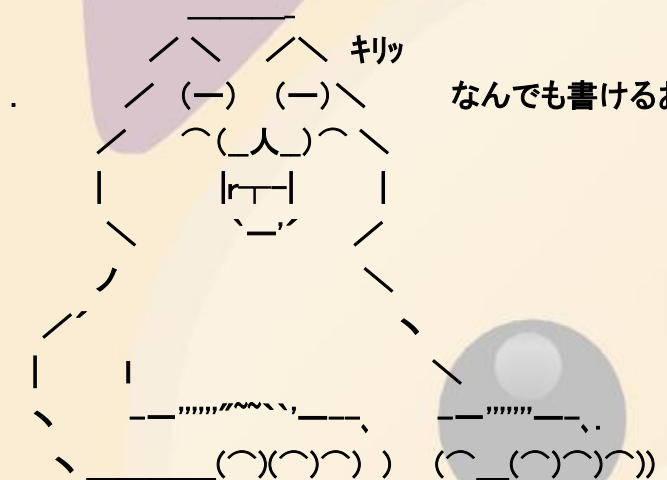


そんな星の元に生まれてるんですよ。

ざんね～ん。

先日、スタッフから言われました。_|￣|○

言語は何ができますか？



よく、聞かれます

CISC/RISC アセンブラ
COBOL
Pascal
BASIC, Visual BASIC
C
C++
Perl
Java

言語にこだわらない

コンピュータにできること＝
CPUとメモリと周辺機器でのやりとり。
言語は違っていても、できることは限られているし、おんなじ。

思想の違い　　：手続き型言語、オブジェクト指向言語

ライブラリの違い：標準ライブラリ、クラスライブラリ、
フレームワーク、etc…

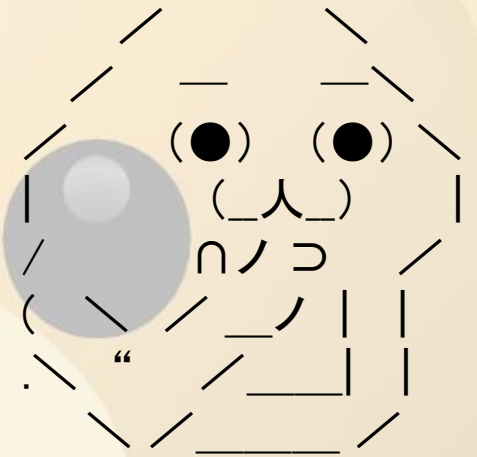
そろそろ本題

匠の伝承

そんなことないんだお。

本日覚えて帰っていただきます。

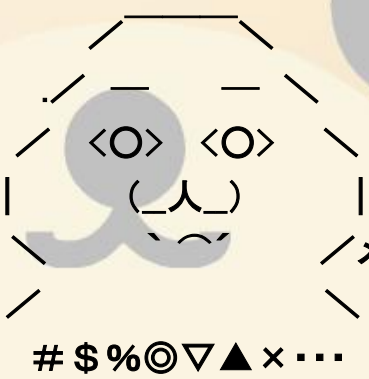
- オブジェクト指向のとらえかた
- 状態のとらえかた



最初から知ってるよ･･･ってひとは、再認識ってことで。 m(_._)m

オブジェクト指向ってなんだ？

継承　クラス　ポリモーフィズム　バーチャル関数
メソッド　カプセル化　デザイン・パターン
多態性　OOP　インスタンス
Java　C++　CLI　UML　アクティビティ図
汎化／特化　オートマトン
多重継承　集約　属性　メンバー変数
コラボレーション図
基底クラス　情報隠蔽　関連　内包
イベント
仮想関数　ステートマシン　派生クラス



is-a と has-a　プロパティ　抽象モデリング
インスタンス　サブクラス　メタクラス
アジャイル開発　ユビキタス
＃＄％◎▽▲×・・・
ソフトウェア部品化　再利用　コンポーネント

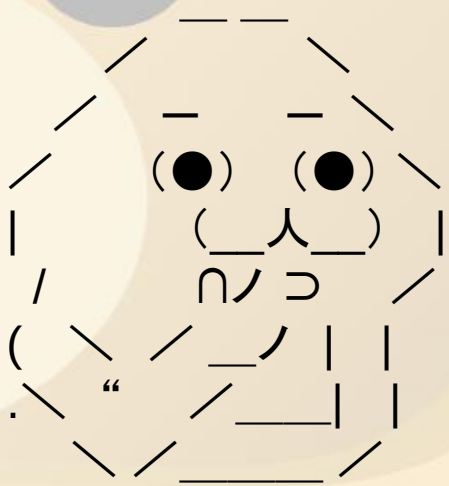
それぞれのオブジェクト指向

実案件から考察してみよう

客先からの要件

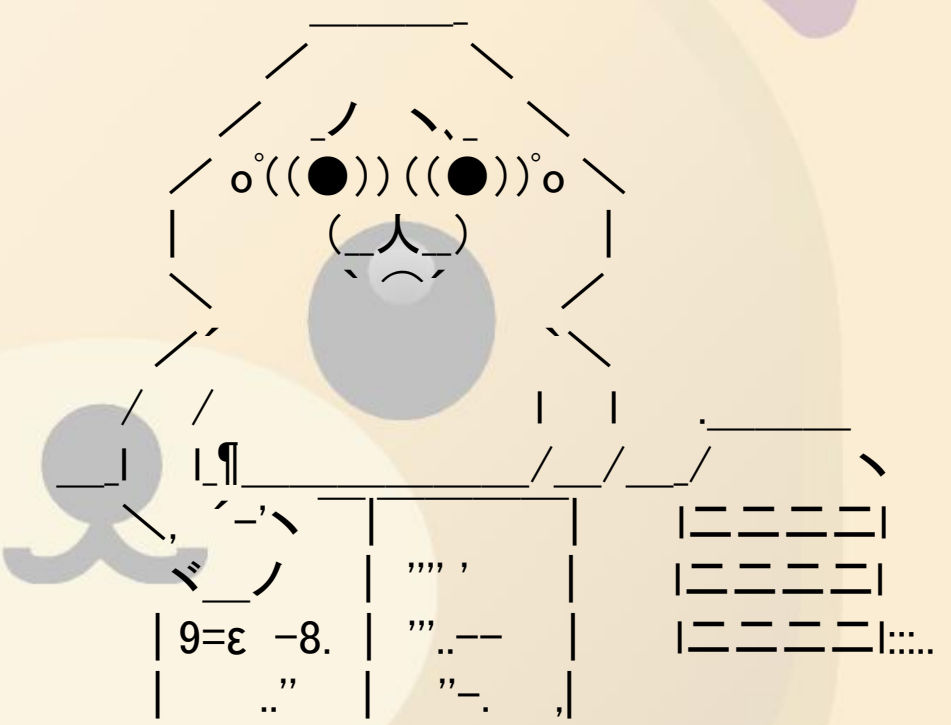
1. USBメモリの製造工程で製品検査
2. 転送速度が規定値の範囲内かを検査
3. 検査が完了したら抜き差しで次の製品

予算少ないから、UIとか設定とかいらないから。
ちょちょいって作るだけでイイから。
ね！？おねがい。

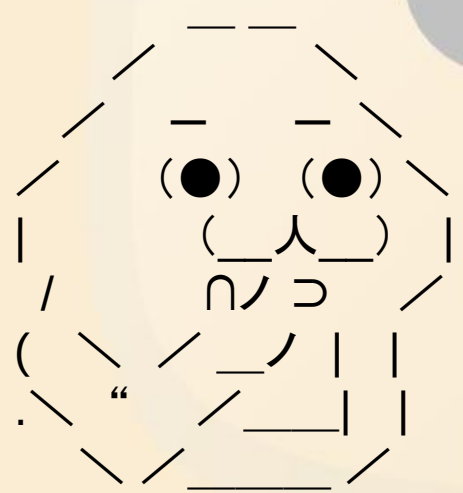


技術的な課題解決が先行する

- USBメモリの抜き差しをどうやって検出するんだろう？

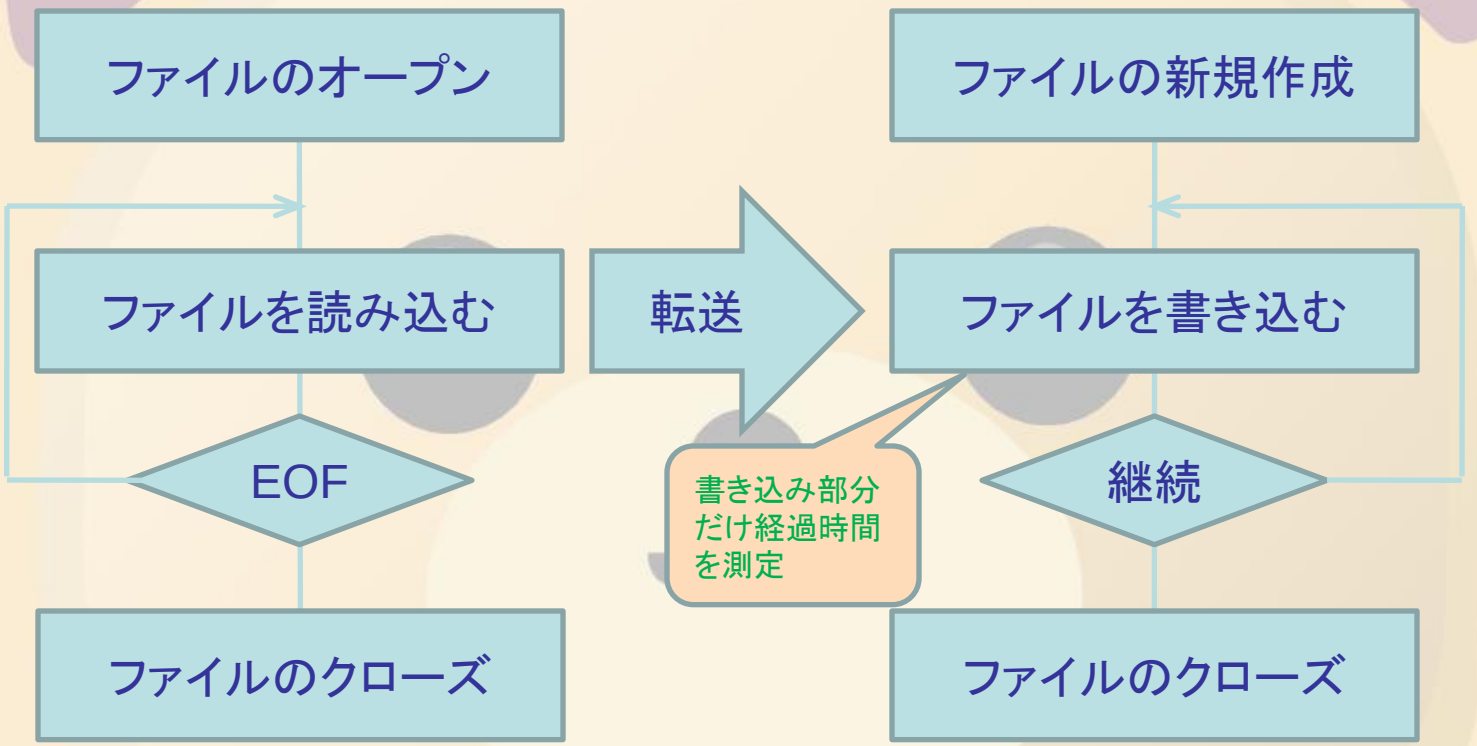


- 転送時間を計るって？



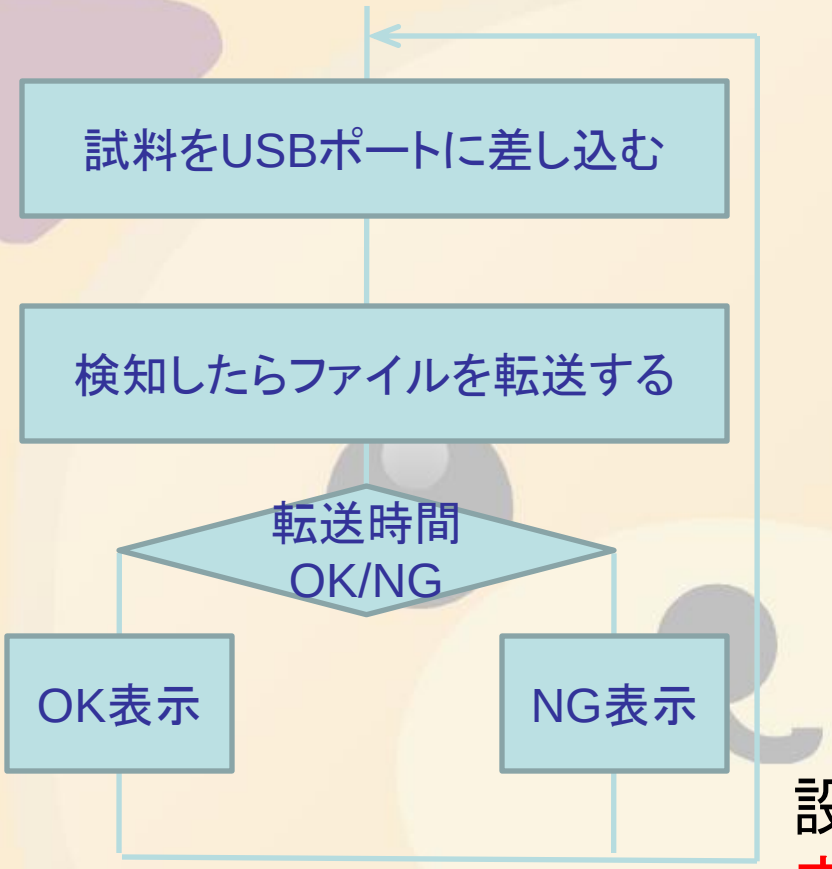
フロー（処理の流れ）を検討する

- おおきなファイルを転送する



※ファイルの転送では厳密な転送速度を測定できませんが、この業務ではある程度の性能が測定できればOKでした。

- 製品の検査の流れ

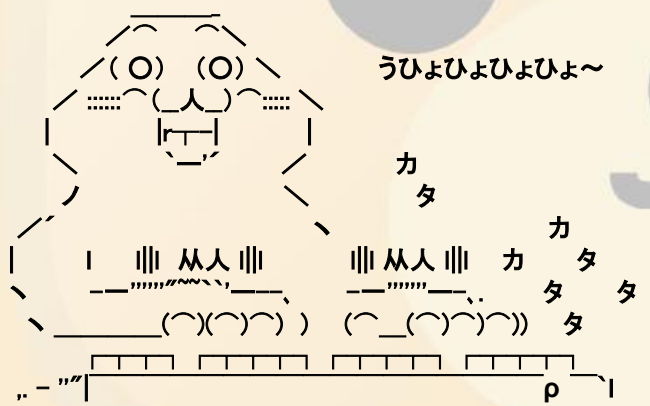


設計／開発者の思考は、
内部処理の実現手続きから
始まっていく

設計すっとばしてすぐ開発

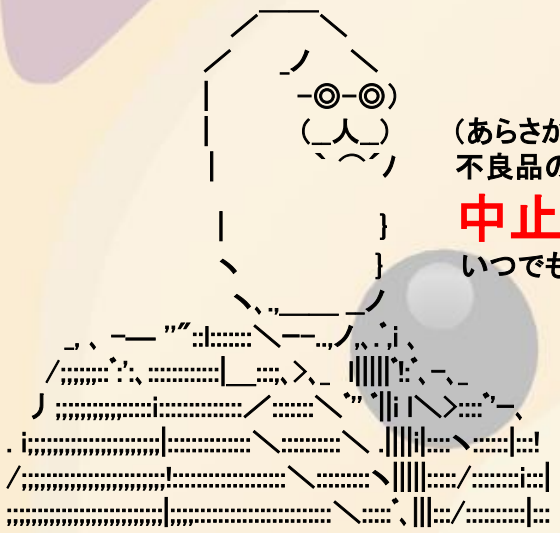
```
handle = Open( "･･･", .. );
Read( handle, buffer, .. );
Write( handle, .. );
Close( handle );
```

…

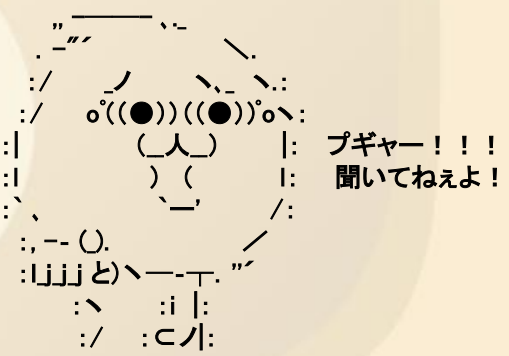


ボトムアップ開発

追加要件に対応しづらい
再利用しづらい



（あらさがし、あらさがし・・・）
不良品のUSBメモリをさすと、ダンマリになりますね。
中止ボタンを出しておいて
いつでも中止できるようにしてください。

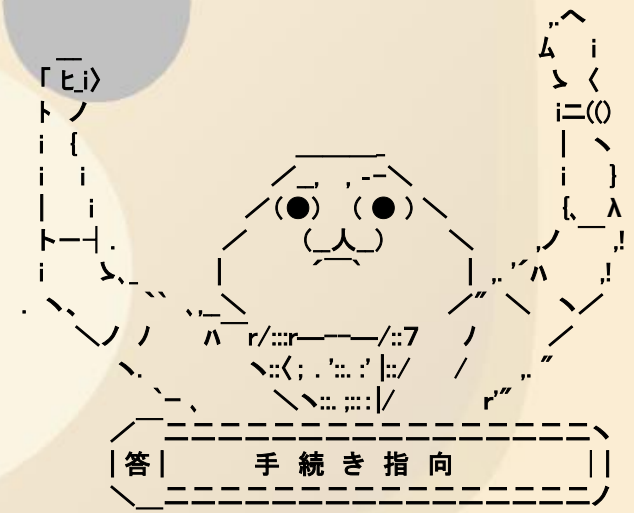


手戻りが多く発生しやすい

あ。納期2日ね。

（がたがた言ってると、もう仕事 流さねぇぞ！）
動いてんのか、死んでんのかわかりませんね。

プログレスバーを出すように
してください。



実現手段

「手続き＝手段」

開発フェーズ

「視点をモノに注目」

目的

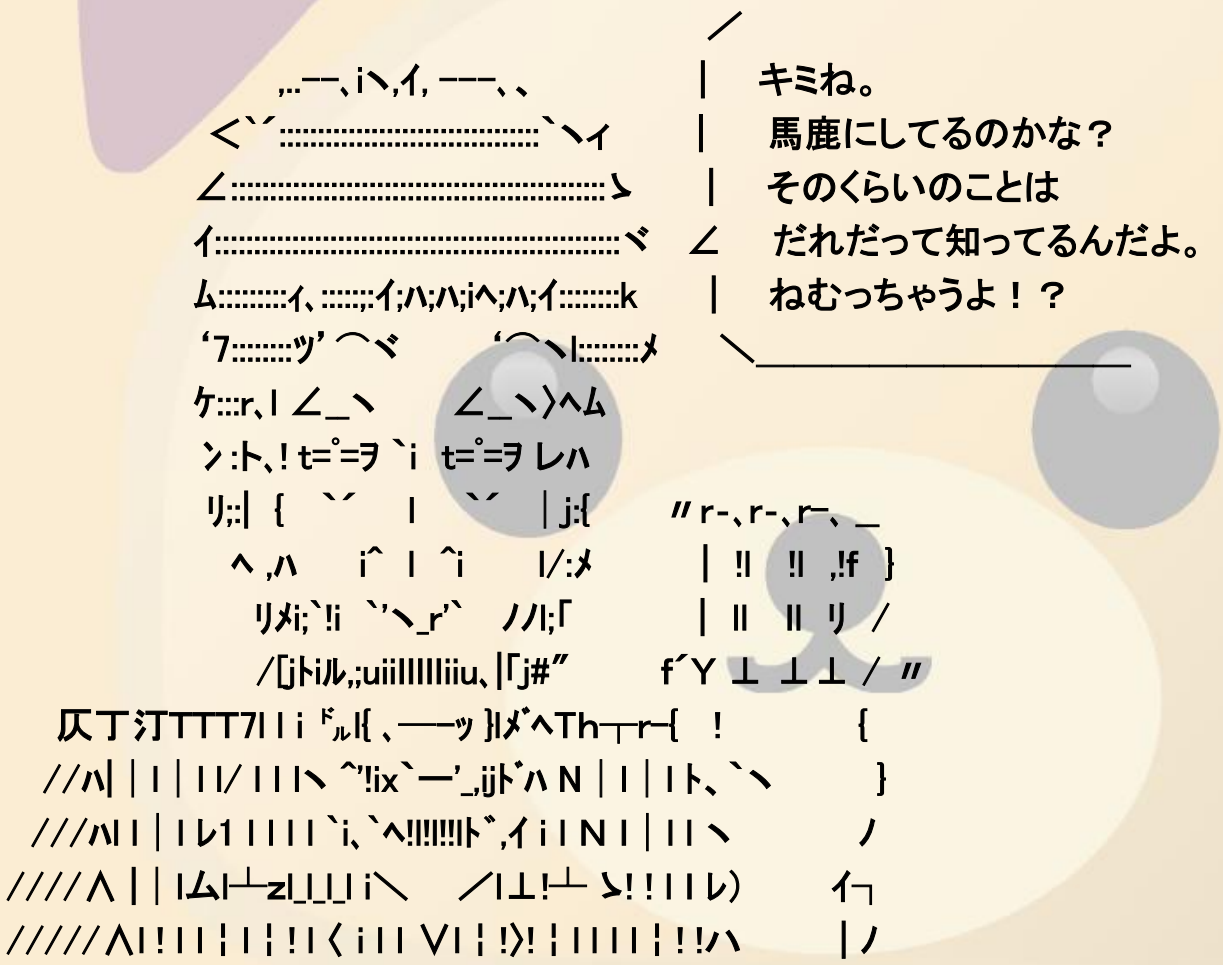
設計フェーズ

オブジェクト指向

設計フェーズで「モノ」をとらえる

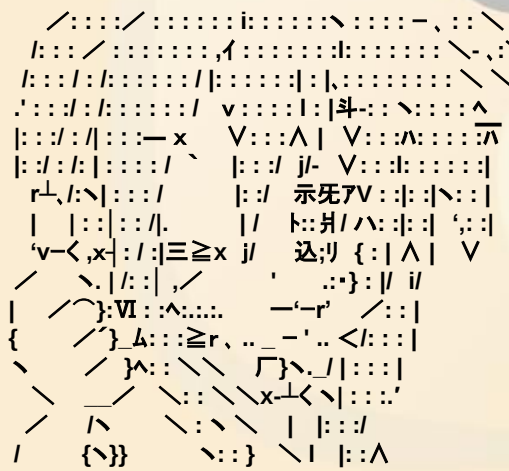
- すでにあるモノ。
  - 「パソコン」（そこにある）
  - 「OS」（そこにある）
- 要件に必要なモノ。
  - 「USBメモリ」（用意できる）
  - 「でっかいファイル」（用意できる）
- これから形になるモノ。
  - 「プログラム」（まだできていない）

ヽ(*`Д´)ノ



実際に携わった仕事の例

- 通信クラス
  - RS-232Cのシリアル通信で特殊な機器とやりとりをさせようと作られていたクラス。
    特に違和感も問題もなく使えました。
- 保守クラス
- 運用クラス
  - 通常運用ではなく、特殊な起動方法やコマンドを入力することでログの参照や特殊な処理が実行できるようになる保守モードを持っていました。
    その2つを区別してクラス設計されていました。



なにがいけないのかわかるかな～？

実際に困った。

システム

同一機器に対して2つのインスタンスを介して処理を呼び出さなければならない

運用クラス
インスタンス

保守クラス
インスタンス

ターゲット機器

同時に通信できるのは一方のインスタンスのみ。

動詞をクラス名にしない

- サ変動詞かどうかを必ずチェック！
  「～する」という言葉を付けてみる

  通信する。
  運用する。
  保守する。

- 形容詞をクラスにとらえられるか？
  「きれい」クラス
  「重い」クラス

プロペラクラスとかも話題になってたなぁ

オブジェクト指向開発の入り口

モノをとらえた設計になっているか

「モノ」＝名詞

【重要】

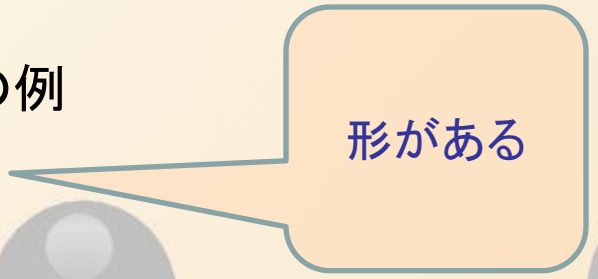
抽象化モデリング

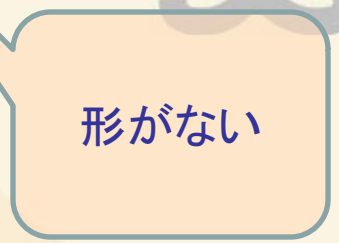
「目に映るモノから」クラス化していく

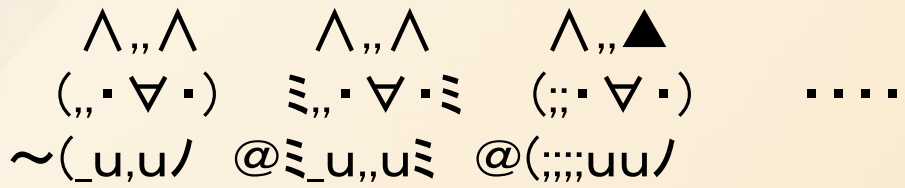
業務システムの例

「入金伝票」
「出金伝票」
「注文書」

「商品コード」
「商品名」

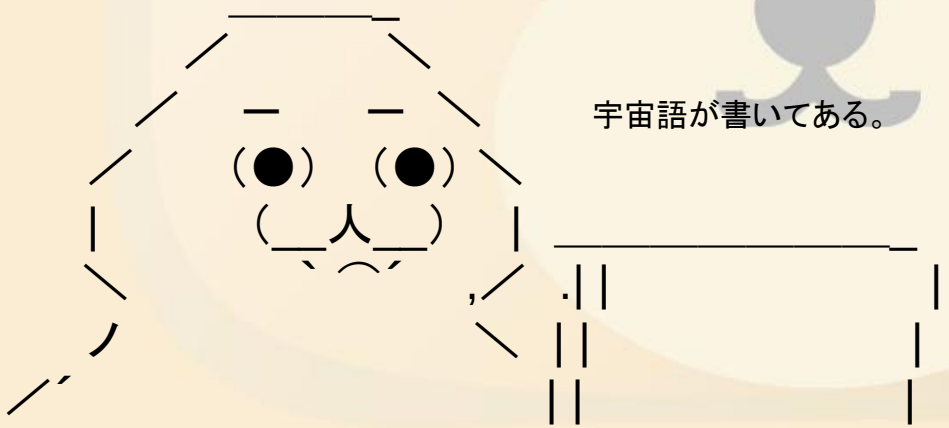




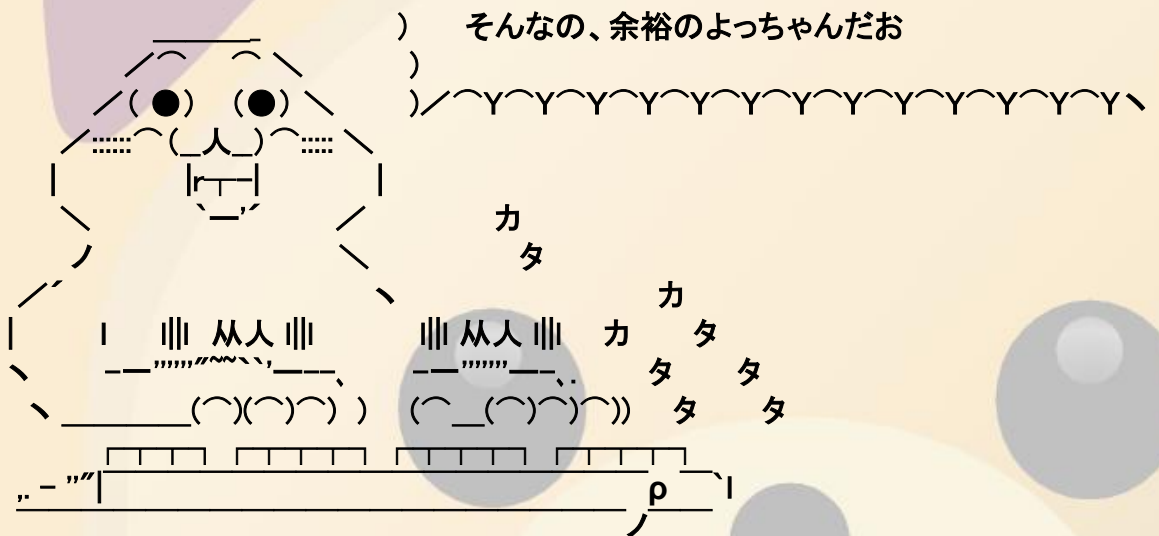

∧,,∧
(,,・∀・)
〜(_u,uノ
∧,,∧
ミ,,・∀・ミ
@ミ_u,,uミ
∧,,▲
(;;・∀・)
@(;;;;uuノ
・・・・

状態を管理する

- ステートマシン
- オートマトン
- ステートパターン
- 状態遷移図
- イベントトレース図
- 状態遷移表



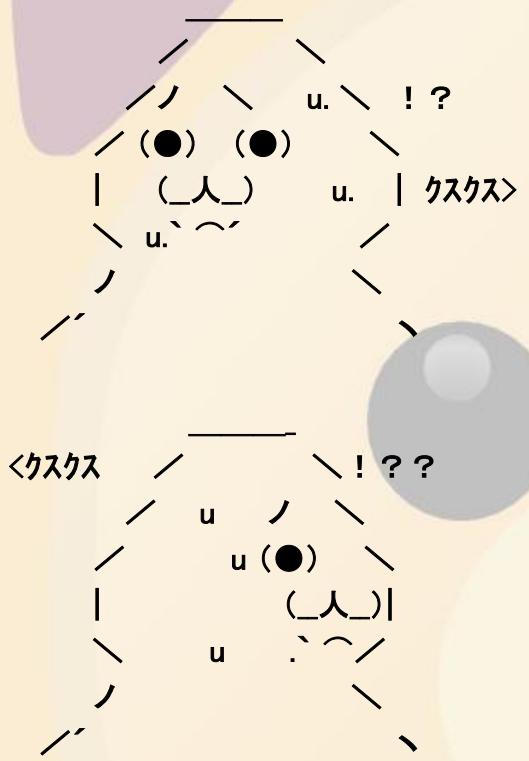
USB検査プログラムの状態とは？



- USBメモリがさされるのを待っている状態（挿入待ち）
- USBメモリにファイルを転送している状態（検査中）

この2つかな？

あってます？

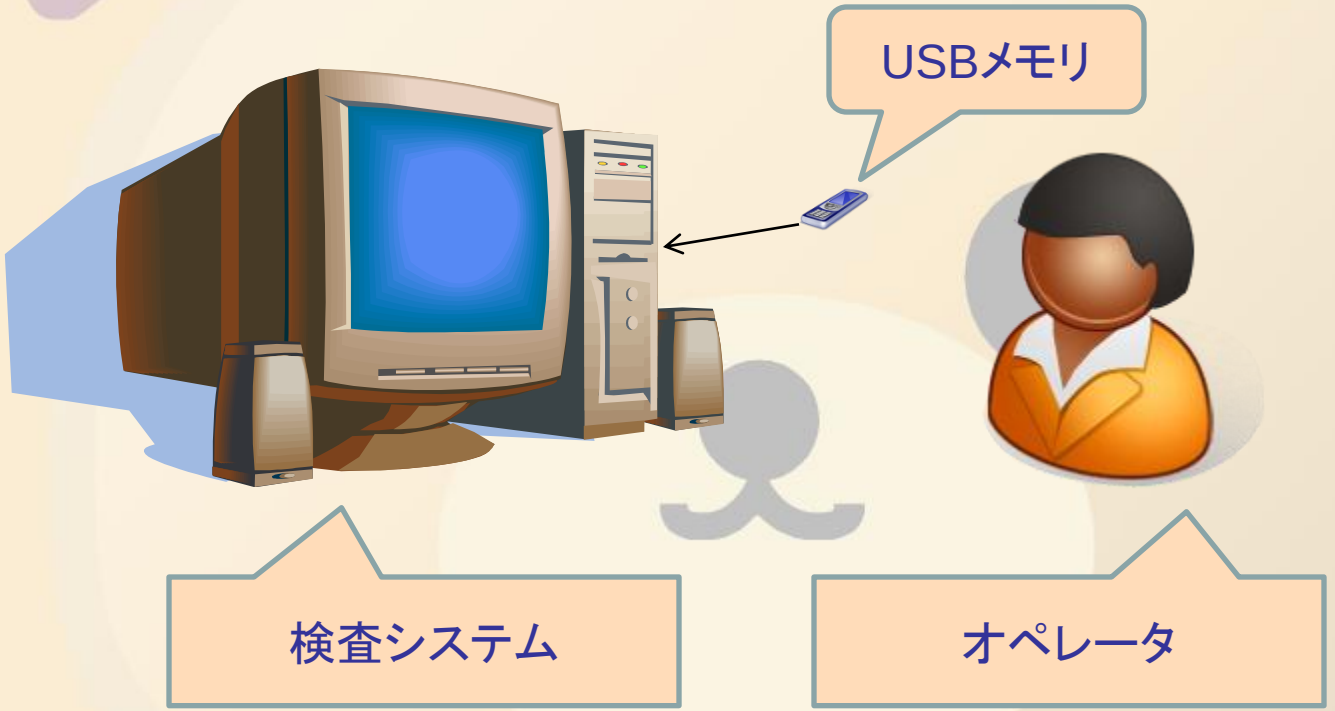


状態と遷移の把握が重要

どうやって状態を切り出す？

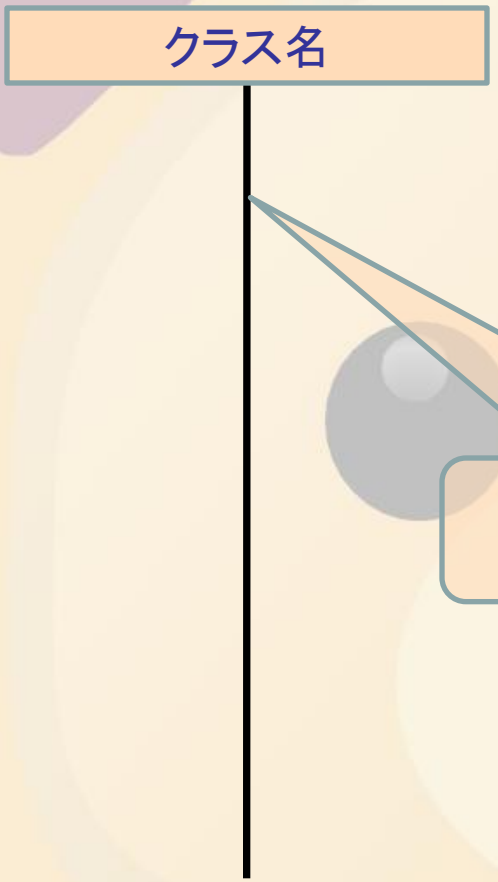
どのように考えるか？

まず、「モノ」の動きをとらえる。



動きを表現する入口はイベント・トレース図

イベント・トレース図を描く



1. クラス名を先頭に書き、縦に線を引く。
2. それだけ。

モノとしてとらえたクラスです

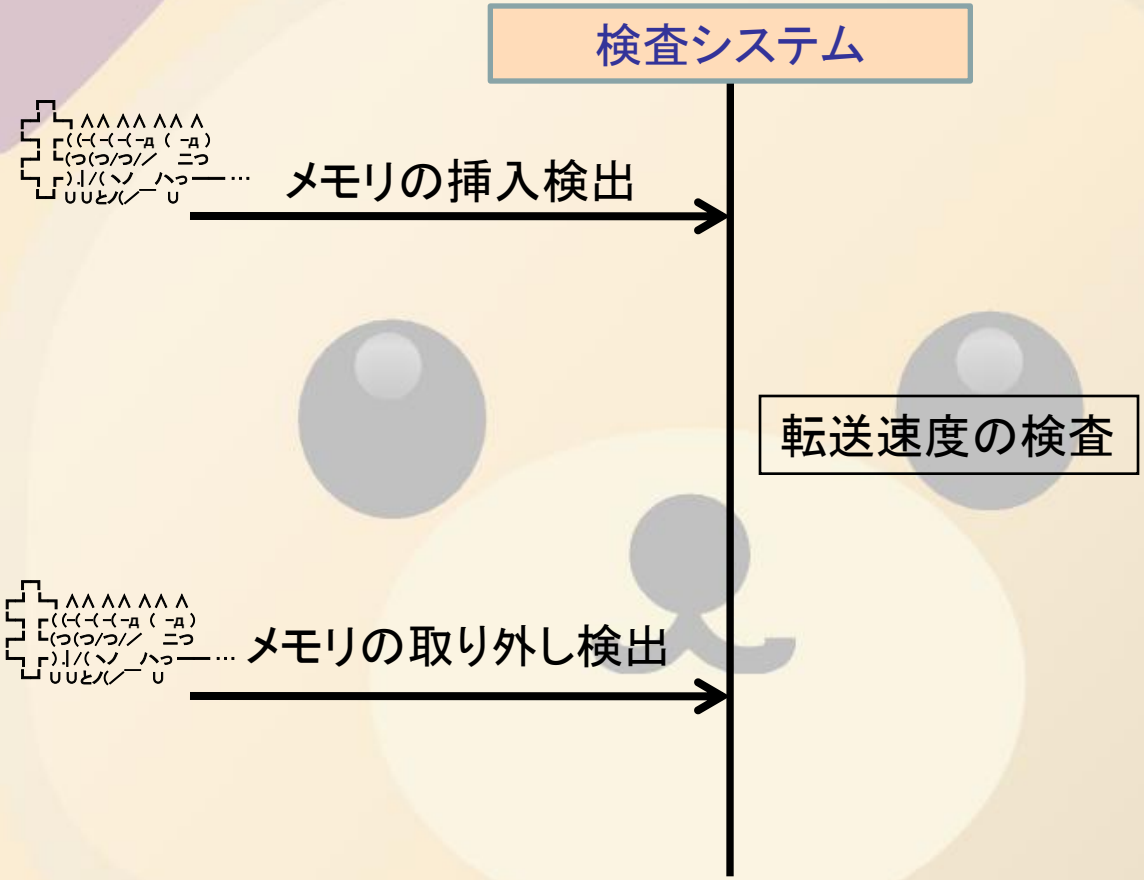
縦線は時間軸を表します

実際に書いてみる

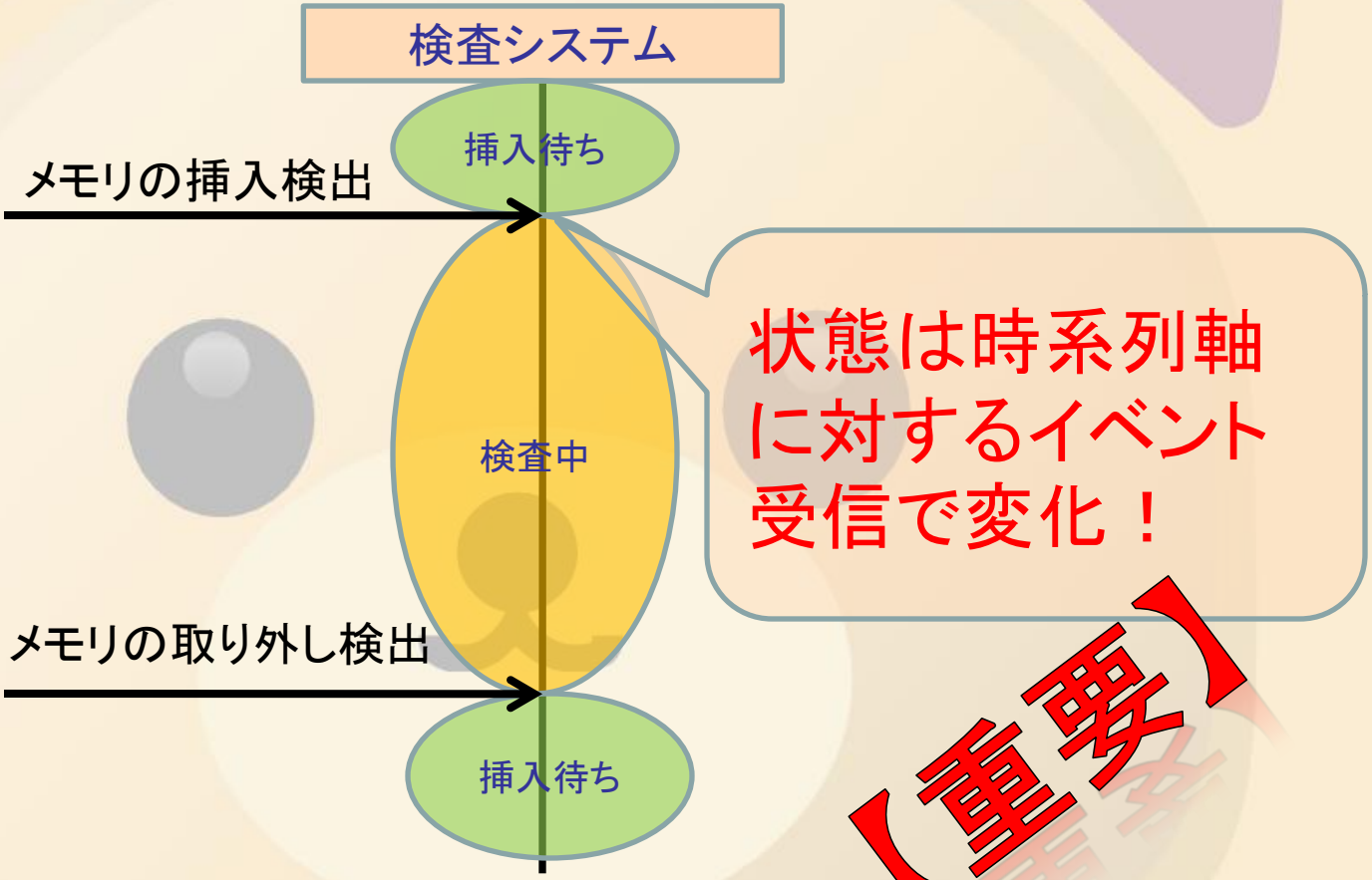
検査システム

いっぽん！？

発生事象(イベント)を時系列順に書く



状態の区別の仕方



状態＝オブジェクトのイベント間を示す

状態遷移表の見方

横軸タイトルは、現在の状態

状態 イベント	状態名 （状態番号）	状態名B （状態番号）	状態名C （状態番号）
イベント1	処理内容 →次の状態	処理内容 →次の状態C	処理内容 →次の状態
イベント	処理内容 →次の状態	処理内容 →次の状態	処理内容 →次の状態
：			

縦軸タイトルは、発生する事象

現在の状態がBで、イベント1が発生したときの処理の内容とその後の状態を記す。

実際に書いてみる

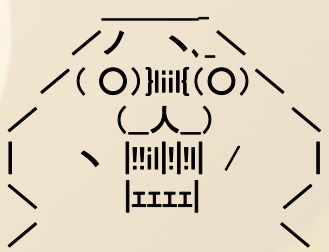
「メモリ挿入待ち」状態でメモリの挿入を検出したら、検査を開始する。
状態は「検査中」になる。

1）検査中にメモリが挿入されたらどうなるの？

状態 イベント	メモリ挿入待ち (0)	検査中 (1)
メモリ挿入検出	検査を開始。 状態を検査中に→(1)	
メモリ取り外し検出		

2）挿入待ちの時に取り外し検出が来たら？

3）検査中にメモリが取り外されたらどうなる？

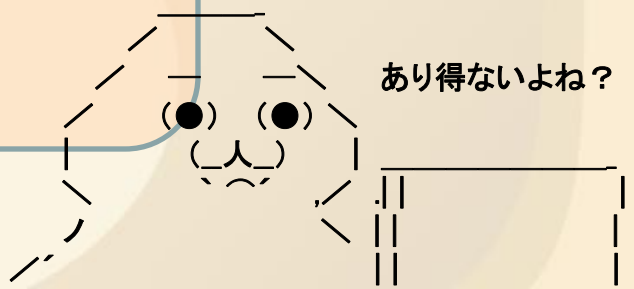


×あり得ないよね。

実際には組み合わせとして、発生することのない状態下のイベント発生を、「あり得ないよね！」といって書かない（該当箇所の対応コードを記述しない）ケースが多い。

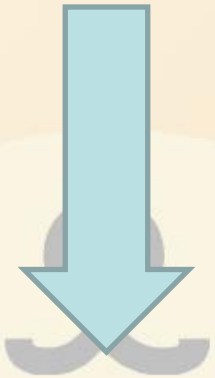
物理的な事象ではなく、ソフトウエアの中身は数値化された値の組み合わせであり、バグや電子的な問題でそういった組み合わせになることがありえる（可能性がゼロではない）。

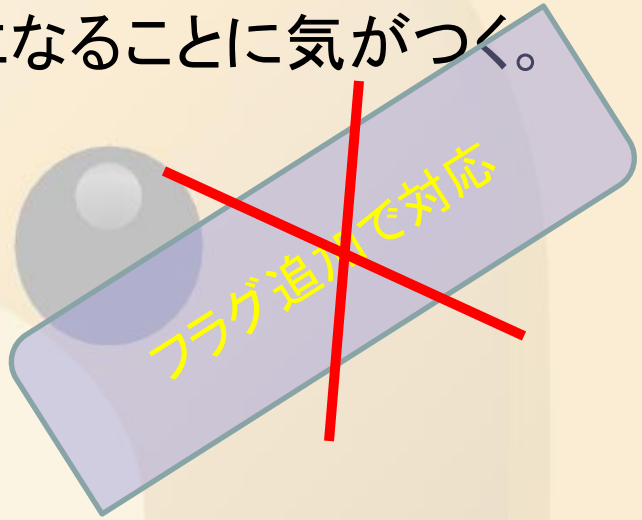
バグの温床



×フラグだよね。

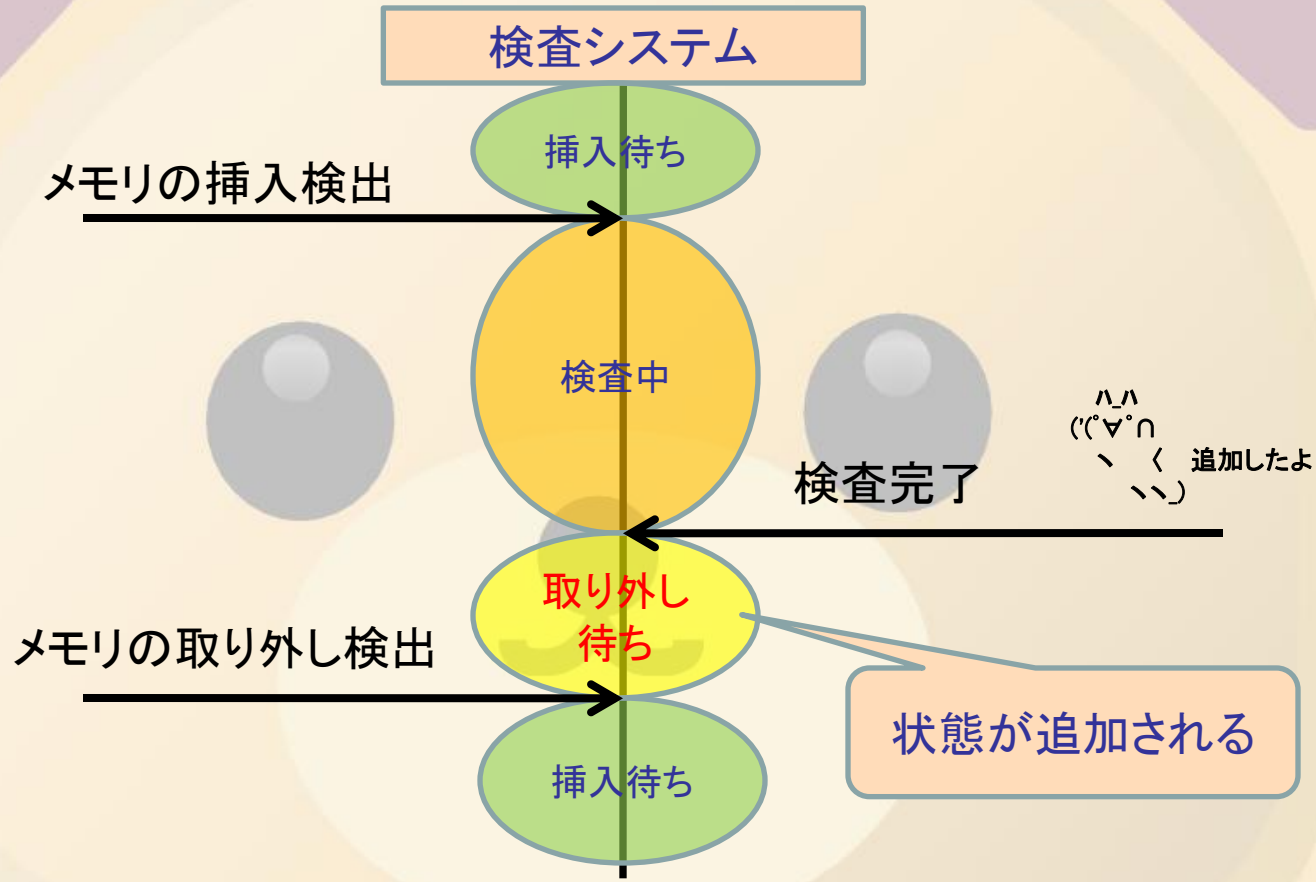
検査中にメモリが抜き出されるのではなく、検査完了時にメモリを抜いても良い状態になることに気がつく。





状態の追加＝イベントの追加
※状態の追加（マトリクスの肥大化）を躊躇しない

イベントの追加と状態の追加



状態遷移表を更新

状態 イベント	メモリ挿入待ち (0)	検査中 (1)	取り外し待ち (2)
メモリ挿入検出	検査を開始 状態を検査中に →(1)	エラー表示 検査中断 →(2)	ログ記録 検査を開始 →(1)
検査完了通知	ログ記録 →(0)	検査結果を表示 →(2)	ログ記録 →(2)
メモリ取り外し検出	ログを記録 →(0)	エラー表示 検査中断 →(0)	次の検査準備 →(0)

正常なシーケンス

事故につながったりします。

マトリクスをすべて埋めること

実際のコードって、どう書く？

パターン1.「イベント」別にモジュールの入口を用意する？

```
メモリ挿入検出処理() {
  if(現在の状態が「メモリ挿入待ち」) {
    検査を開始()
    状態を「検査中」に
  } else if(現在の状態が「検査中」) {
    エラー表示()
    検査中断→状態を「取り出し待ち」に
  } else if(現在の状態が「取り出し待ち」) {
    ログ記録()
    検査を開始()
    状態を「検査中」に
  }
}
```

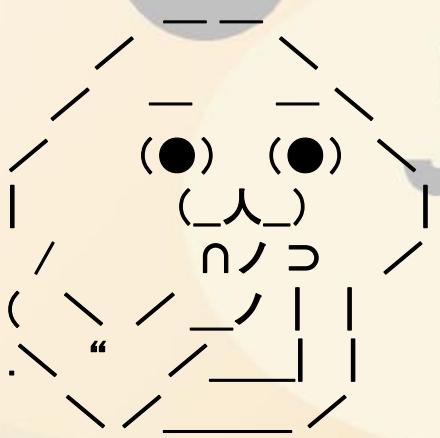
多くの開発ツールで、ボタンイベントに対応するコードなどが、こういった事象の発生箇所を入口にしてコードを書くようになっていますね。

OnButtonClick(),
OnInitDialog()
ON_WM_PAINT(),

パターン2.「状態」別にモジュールの入口を用意する？

```
メモリ挿入待ち::メモリ挿入検出処理() {
        検査を開始()
        状態を「検査中」に
}
メモリ挿入待ち::検査完了通知(){
        ログ記録()
}
メモリ挿入待ち::メモリ取り外し検出(){
{
        ログ記録()
}
```

この形式にするには、状態ごとに切り分けられた単位を用意する必要がありますねぇ・・・



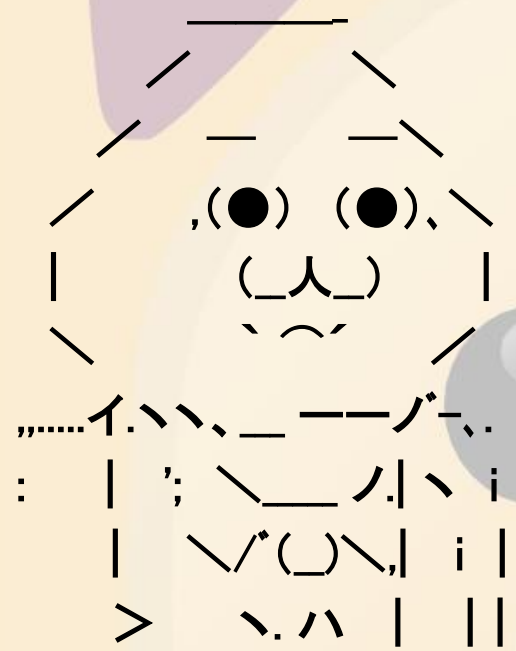
どっちにしたらいいのかな？

実装は状態別に

状態 イベント	メモリ挿入待ち (0)	検査中 (1)	取り外し待ち (2)
メモリ挿入検出	検査を開始 状態を検査中に →(1)	エラー表示 検査中断 →(2)	ログ記録 検査を開始 →(1)
検査完了通知	ログ記録 →(0)	検査結果を表示 →(2)	ログ記録 →(2)
メモリ取り外し検出	ログを記録 →(0)	エラー表示 検査中断 →(0)	次の検査準備 →(0)

状態別にイベント発生時の処理を書く

ステートパターン．．．



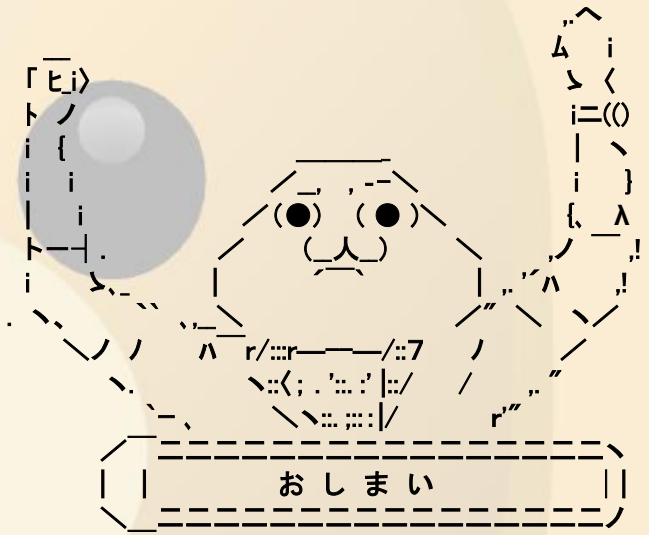
残念ながら、時間が来たようです。
続きは、また別の機会にでも。

こっからが肝心だお

また今度だお

ご静聴ありがとうございました。

m(_._)m



伝承でもナンでもなかった件について（w

Special thanks for Yaruo charactors