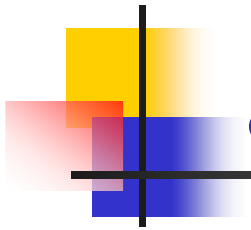
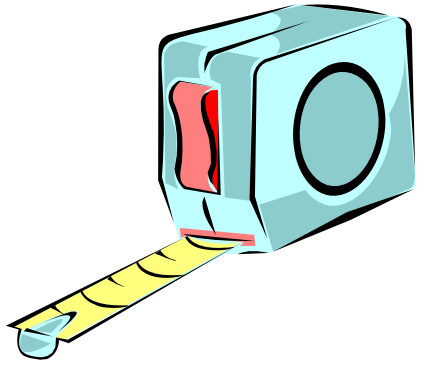


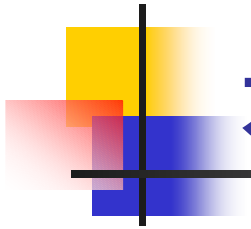
よい設計を行うには？

- 新規によい設計を行うのは難しい
- 既存の設計の修正
- どこを直せばよいか
  - ソフトウェアメトリックス
    - ソフトウェアの数値化．複雑度など
  - リファクタリング：Refactoring
    - 振る舞いを変えることなく，ソフトウェアの内部構造を変える．
    - 理解や変更が容易になるように，

ソフトウェアメトリックス

- よい設計・ソフトウェアとは？
- ソフトウェアの値段は？
- 開発にどれぐらいかかる（期間）？
- 計測：ソフトウェアを数値化する
  - 例：行数（プログラムの大きさ）
- どのような観点から？
  - 品質のメトリックス
  - 複雑さのメトリックス

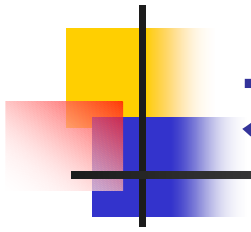




オブジェクト指向のメトリックス（１）

オブジェクト指向設計・プログラムの構成要素

1. クラス
2. メソッド
3. 属性（インスタンス変数）
4. メッセージ通信
5. 関連
6. 継承関係



オブジェクト指向のメトリックス（2）

1つのクラスに対して

1. WMC (Weighted Methods per Class)
   そのクラスが持っているメソッド数
2. DIT (Depth of Inheritance Tree)
   根からそのクラスまでの継承木の深さ
3. NOC (Number of Children)
   直属のサブクラス数
4. CBO (Coupling between Object Classes)
   そのクラスが参照するインスタンス変数やメソッドを持っているクラス数

オブジェクト指向のメトリックス（3）

5. RFC (Response for a Class)

そのクラスのインスタンスがメッセージを受信したときに実行される可能性のあるメソッド数

クラスに含まれるメソッド数＋呼び出されるメソッド数

6. LCOM (Lack of Cohesion in Methods)

メソッドで共通に使用されるインスタンス変数がないようなメソッドの度合い

Ii：メソッドMiが使用するインスタンス変数の集合

$P = \{(i, j) \mid Ii \cap Ij = \Phi\}$　$Q = \{(i, j) \mid Ii \cap Ij \neq \Phi\}$

$$LCOM = \begin{cases} |P| - |Q| & |P| > |Q| \text{ のとき} \\ 0 & \text{上記以外} \end{cases}$$

例