目的・範囲: *Objective & Scope*	Design Pattern(ChainOfResponsibilityパターン)
分類名: *Classification Name*	
著作者: *Author*	トショウケイ
実施日: *Enforcement day*	2014.05.17
バージョン: *Version*	1
初版発行日: *Original Release*	
現行版発行日: *Current Release*	
キーワード: *Key Words*	
背景情報: *Background*	

◇目的：処理の責務をたらい回しにする。口

◇効果：利点としては
・処理の連鎖の修正が容易
・各自の処理に専念できる
・処理のすべてをクライアントが知る必要がないということは、処理するオブジェクトのメンテナンス性を高めることにもつながります。

◇背景：複数のオブジェクトを鎖（Chain）のようにつなぎ、その鎖を渡り歩いて目的のオブジェクトを決定する
窓口のたらい回し
このパターンでないと、処理の割り振りを行う中央集権的オブジェクトが必要になる
Client に担当させるのはよろしくない
要求側と処理側の結びつきを弱められる
双方の独立性を高められる
動的に連鎖形態を変化することができる
GUI ウインドウにコンポーネントを配置する場合
ConcreteHandler は自分の仕事のみに集中できる
処理できなかったらあっさり次の ConcreteHandler に渡す
◇ChainOfResponsibilityパターンの実際のコードと考え方
サンプルのファイルご覧ください

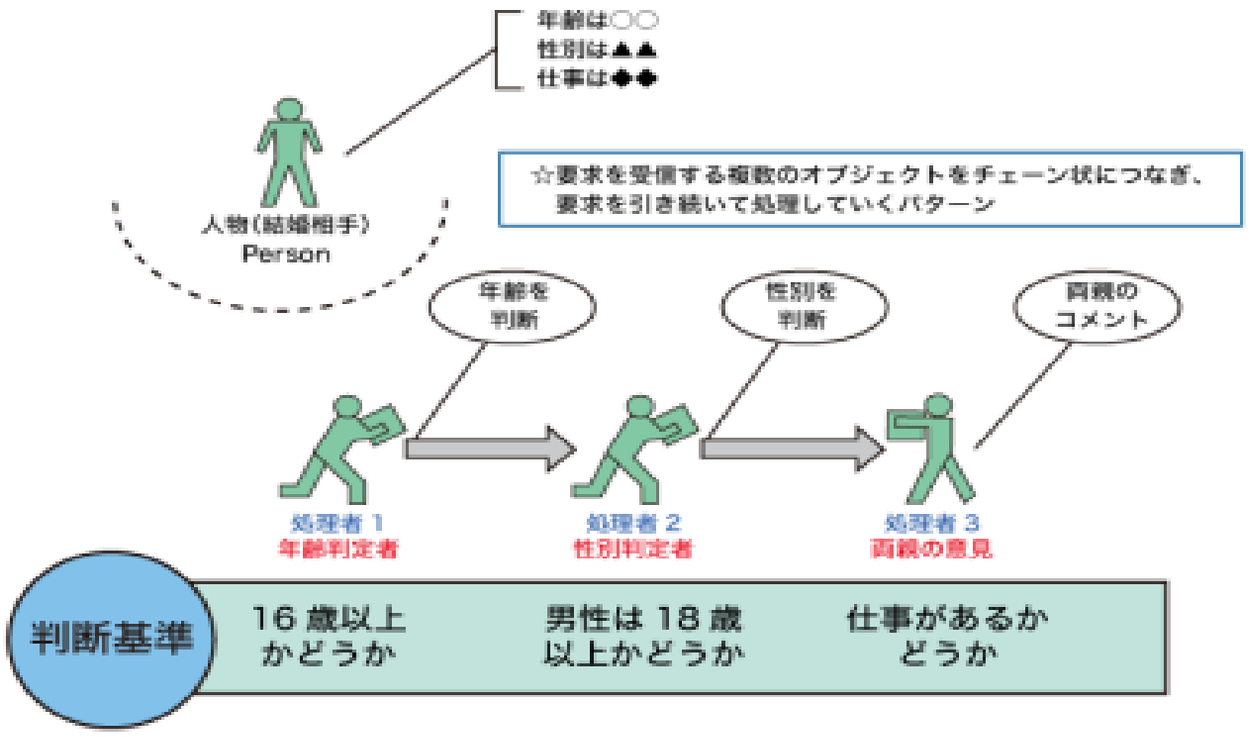


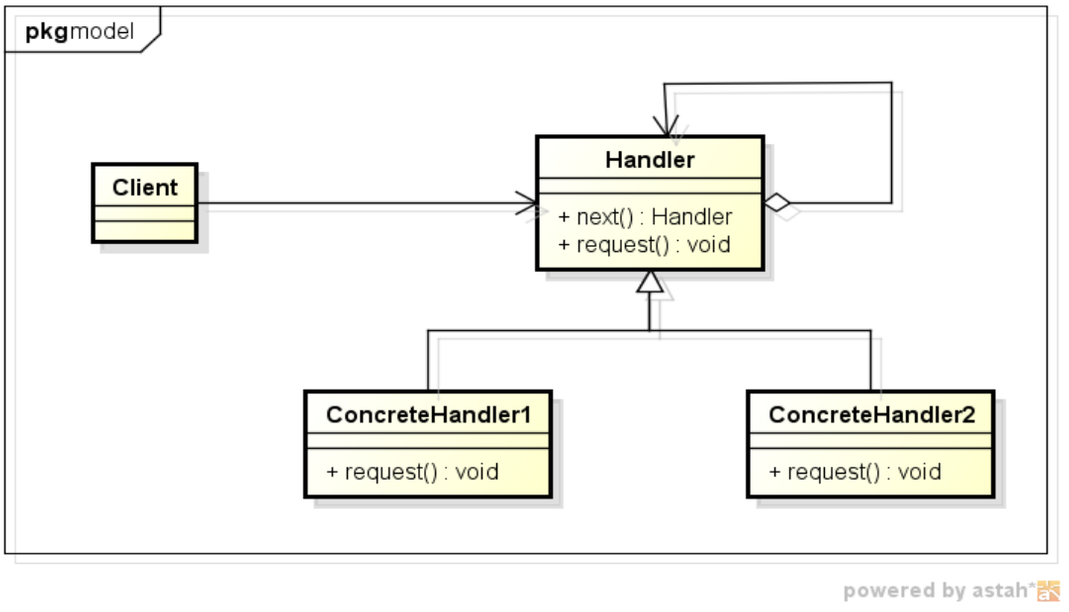
図２：Chain Of Responsibility イメージ図　結婚判定ワークフロー

・対象者が16歳以上かどうか
・性別を判定して男子であれば18歳以上かどうか
・仕事を持っているかどうか

この3つの処理を順次オブジェクトに引き継ぎ処理をしていきます。

◇問題点の改善：
・欠点としては処理の遅いこと

◇ChainOfResponsibilitパターンのまとめ



Chain of Responsibilityパターンは、要求に対する処理が完了するまで、要求を受信する複数のオブジェクトをチェーン状につなぎ、要求を引き続いて処理していくパターンです。言い換えると、ある処理を行うのに、その特定できるオブジェクトが不明な場合、現在処理しているオブジェクトに次の処理者を登録して、チェーンのようにつなぎながら、処理できるオブジェクトに到達するまで、順次要求を渡していく方法です。

◇注意：・チェイン オブ レスポンシビリティは、クラスの責務を分割し鎖のようにつなげるパターンである
・鎖は複雑化しないように1対1対1・・・とつなげる。1対多はコンポジットというパターンでエクスプローラーのようなツ
（ロジックをコンポジットにしたらわけがわからず保守が大変である）
なお、一度つないだクラスを間違って再度つなぐと、鎖の輪っかになって無限ループのスタックエラーになるので注意さ

◇総括

基本用語

用語	説明

※引用文献e-words、WikiPedia