# Ruby の
# オブジェクト指向機能について

中田伸悦
nobu@ruby-lang.org

# Ruby の特徴

「オブジェクト指向スクリプト言語 Ruby 」より

- インタプリタ
- 型宣言が不要
- 書きやすく読みやすい文法
- すべてがオブジェクト
- イテレータ
- 例外処理機能
- …

→ 重要な基本的特徴の一つ

# 概要

- すべての値がオブジェクト
- すべてのクラスは `Object` クラスの子孫
- Mix-in を使った単一継承
- 特異メソッド
- Class-based ではなく Object-oriented

# すべての値がオブジェクト

値 ＝ 操作できる対象

## オブジェクトの例

整数
浮動小数点数
文字列
配列
ハッシュ
正規表現
クラス
モジュール
シンボル

その他…

## オブジェクトでないもの

変数
制御構造

プログラムで表現できないもの

# クラスとモジュール

## メソッドを定義する器

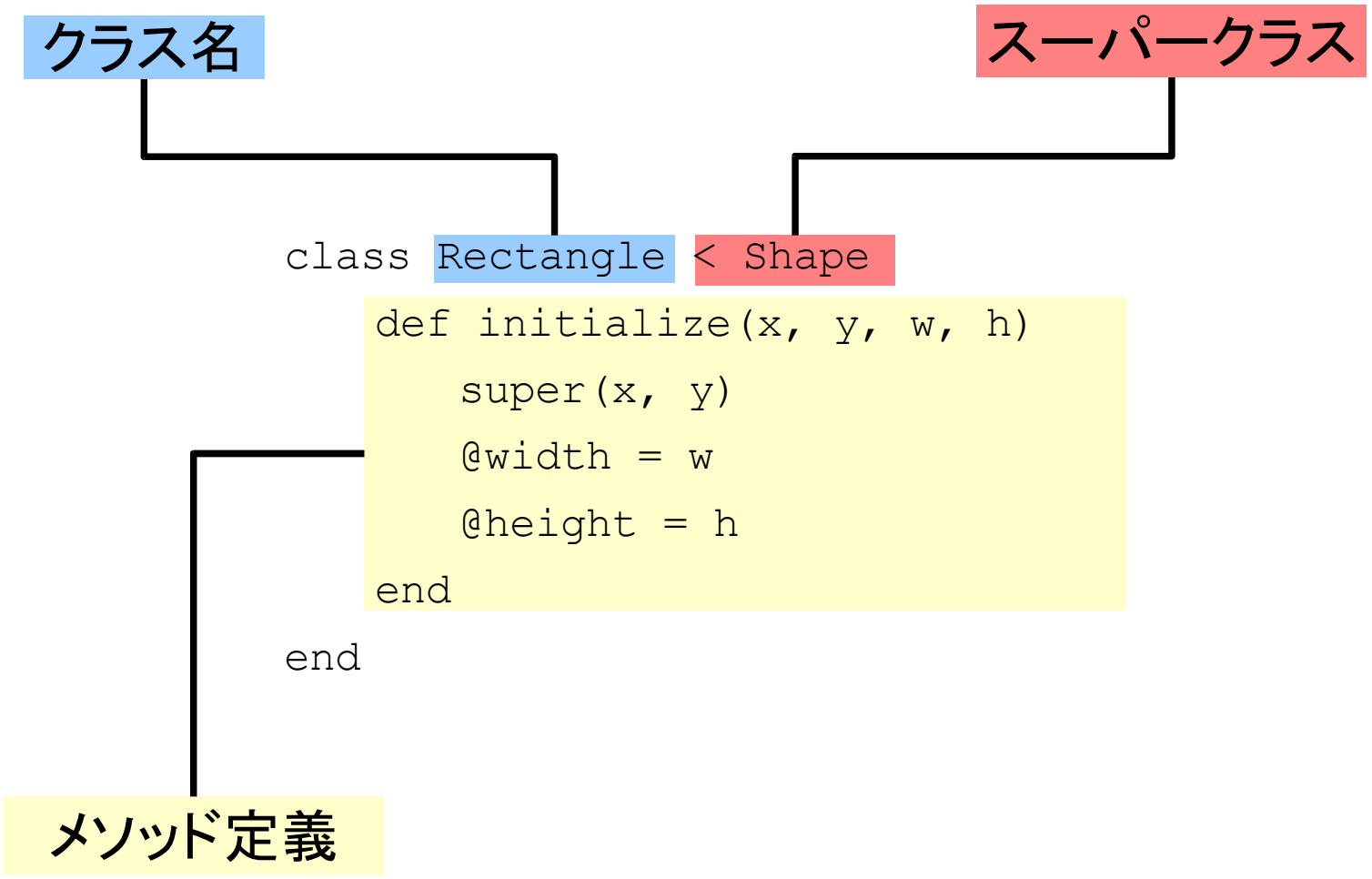
## クラス

- すべてのオブジェクトは何かのクラスに所属する
- C++ や Java のような primitive 型というものは存在しない

## モジュール

- インスタンスは作れない
- 継承できない
- include/extend できる
- 実装を共有できる

# クラス / メソッド定義


クラス名
スーパークラス
class Rectangle < Shape
def initialize(x, y, w, h)
super(x, y)
@width = w
@height = h
end
end
メソッド定義

# 特異クラス / メソッド定義

```
rect = Rectangle.new(0, 0, 200, 100)

class << rect
    def area
        @width * @height
    end
end

def rect.area
    @width * @height
end
```

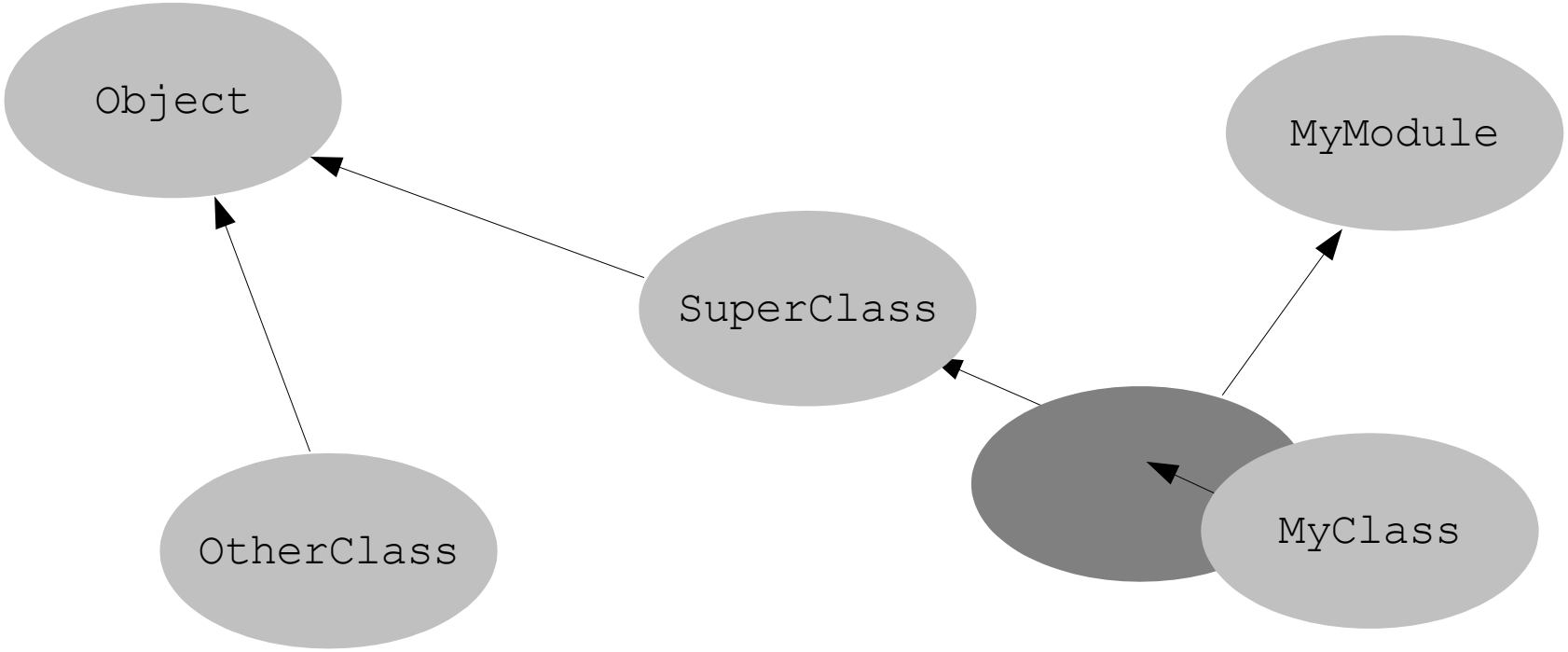
特異クラス

特異メソッド定義

特異メソッド

# オブジェクトの仕組み

- すべてのクラスは `Object` クラスの子孫
- Mix-in を使った単一継承



# デザインパターン

## 特異メソッド / クラスを応用

- Prototype
- Singleton
- Adapter

## クラスオブジェクトを応用

- Factory
- Builder　　例 :Net::HTTP::Proxy
- Bridge

# Ruby にとってのオブジェクト指向

楽しいプログラミングのための

道具の一つ

自然にオブジェクト指向