

# Ruby 初級者向けレッスン 第 14 回

okkez@Ruby関西, サカイ@小波ゼミ

## 自己紹介

---

- okkez（おっきーと読みます）
- 所属は Ruby 関西
- Ruby 歴は二年くらい。もうすぐ三年。
- るりま
- 最近 Ruby 勉強会以外で発表することがありました

## 自己紹介

---

- サカイ
- 小波ゼミの 4 回生です
- Ruby も Java も未熟者です
- 実はピアノが弾けます（10 年やっています）
- 国産の松茸が食べたいです

## 今回の内容

---

- アンケート
- オブジェクト指向
  - 歴史
  - 継承、カプセル化、ポリモルフィズム
- Ruby のコードでは？
- 演習 (group work)
- まとめ

## 今回のゴール

---

- オブジェクト指向の歴史をおおざっぱに知る
- Ruby でのオブジェクト指向をおおざっぱに知る

# アンケート

---

- 構造化プログラミング
- オブジェクト指向
  - なにそれ？おいしいの？
  - 挫折した orz
  - 達人級 (pragmatic)

## 歴史 - オブジェクト指向以前

---

- データ構造もへったくれも無い時代
  - 大昔
  - そこそこの規模(複雑さ)のソフトウェアで破綻
- データと処理を分ける時代
  - 構造化プログラミング → ルーチンの抽象化
  - 大規模なソフトウェアにも対応できる（と言われていた）

## 歴史 - 構造化プログラミングの限界

---

- さらなる大規模(もっと複雑な)ソフトウェアの出現
  - データ構造と処理を分けるだけでは対応不可能に



## 歴史 - オブジェクト指向言語の登場

---

- Simula, Smalltalk
- C++, Java
- Python, javascript
- Ruby

## オブジェクト指向とは？

---

- ソフトウェアの設計や開発において、操作手順よりも操作対象に重点を置く考え方。

## オブジェクト指向とは？

---

- 関連するデータの集合と、それに対する手続き(メソッド)を「オブジェクト」と呼ばれる一つのまとまりとして管理し、その組み合わせによってソフトウェアを構築する。

## 私的オブジェクト指向

---

- 汎用の整理術
- 脳内スタック節約術
- DRY 実現のための考え方
  - Don't Repeat Yourself.

# オブジェクトとは？

---

- 実際にある「もの」や「概念」
  - okkez, サカイさん, okkez のノートPC, などなど
- オブジェクトはデータと処理（メソッド）をセットにしたもの
- オブジェクトは自分ができること（メソッド）を知っている

## クラスとは？

---

- オブジェクトのうち共通の性質を持った「くくり」
  - 人、大学、パソコン、などなど
- インスタンスを作る「雛型」

# オブジェクト指向の三大要素

---

- 継承
- カプセル化
- ポリモルフィズム

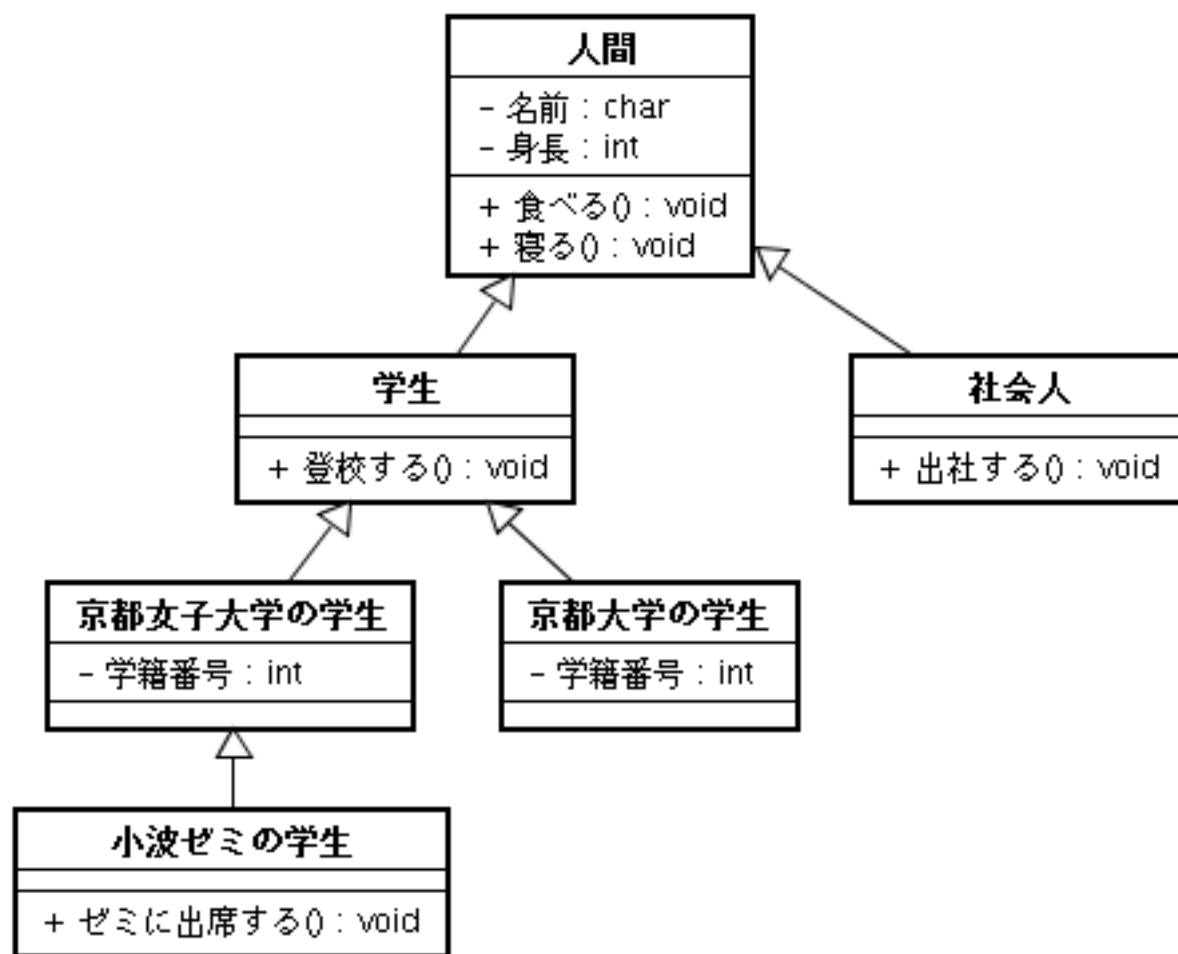
# 継承

---

- 具体的なクラスは一般的なクラスの性質を引き継いでいる
  - 世の中のものをより一般的にくることを抽象化
  - 世の中のものをより具体的にくることを具象化



# 継承



## カプセル化

---

- 仕事（メソッド）のやりかたは担当（クラス）だけが知っていて、仕事を頼む人はその中身は知らない
  - オブジェクト内部の構造は外からはわからない → 抽象データ型
  - オブジェクトの操作はメソッド経由で行う → 変更が強くなる

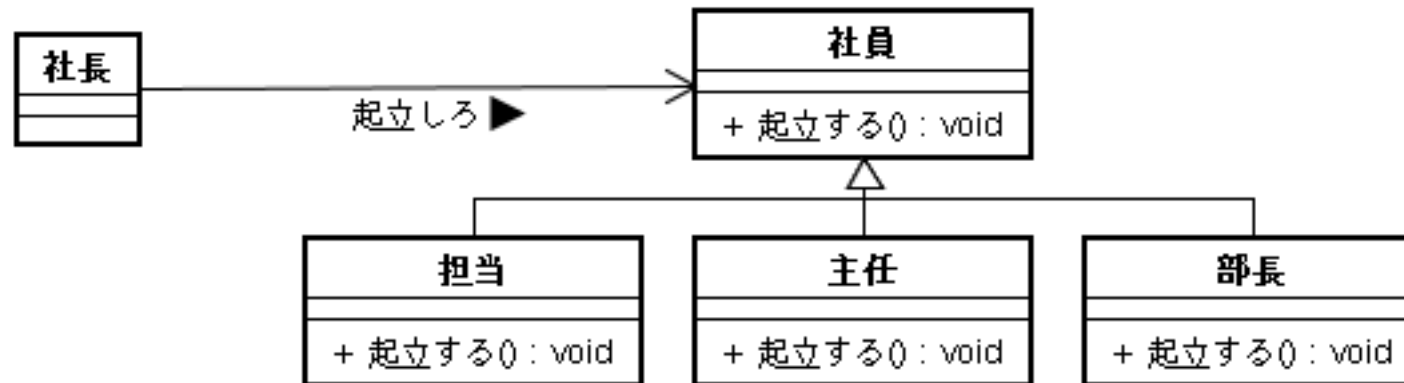
# ポリモルフィズム

---

- 究極奥儀
- 仕事を頼まれる側の種類が増えても、仕事を頼む方法は同じ

# ポリモルフィズム

---



## Ruby での継承

---

- 以下のように書くことができます

```
class SubClass < SuperClass  
end
```

## Ruby での継承

---

```
csv.rb:12:  class IllegalFormatError < RuntimeError;  
            end
```

```
csv.rb:15:  class Cell < String
```

```
csv.rb:26:  class Row < Array
```

```
csv.rb:591: class StringReader < Reader
```

```
csv.rb:615: class IOReader < Reader
```

```
csv.rb:713: class BasicWriter < Writer
```

- この辺に出ています。

# Ruby でのカプセル化

---

- あとで。

# Ruby でのポリモルフィズム

---

- Duck Typing
  - アヒルのように歩き、アヒルのようになくものはアヒルだ



## Ruby でのポリモルフィズム

---

- IO, File, Tempfile, StringIO などなど
  - IO と File は継承関係にある
  - Tempfile は File クラスに処理を委譲している
  - StringIO は IO が持つメソッドを全て実装している

## まとめてコード例

---

- ちょっとだけコード例の説明をします

## 演習1 - 社長命令・起立！

---

- 社長の席のついたての向こうに誰か社員がいます。
- 社長は、社員なら誰でもいい用事を思いだし、声をかけます。
  - 「わしは社長や。誰か知らんけどそこにいる君、立ちなさい」

## 演習1 - 社長命令・起立！

---

- 呼ばれた人はそれぞれなりに起立します。
  - 担当が普通に起立しました。
  - 主任がすばやく立ちました。
  - 部長がだるそうに立ちました。

## 演習1 - 社長命令・起立！

---

- 動作例

\$ ruby shacho1.rb Tanto  
担当が普通に起立しました。

\$ ruby shacho1.rb Shunin  
主任がすばやく立ちました。

\$ ruby shacho1.rb Bucho  
部長がだるそうに立ちました。

## 演習1 - 社長命令・起立！

---

- 社員のコード (shain1.rb) を書きましょ  
う

- Shain クラスを定義し、それを継承して Tanto, Shunin, Bucho クラスを作ります。

```
class Shain
  def standup
  end
end
```

```
class Tanto < Shain
  ...
end
```

## 演習1 - 社長命令・起立！

---

- shain1.rb (一部)

```
class Tanto < Shain
  def standup
    puts("担当が普通に起立しました。")
  end
end
```

```
class Shunin < Shain
  def standup
    puts("主任がすばやく立ちました。")
  end
end
```

## 演習2 - 給料はいくら？

---

- 社長からさらに命令が出ました。
  - 「誰か知らんけど基本給を教えるから、そこから計算して君の給料がいくらか答えなさい」
- 給料計算のルール
  - 担当：基本給と同じ
  - 主任：基本給 \* 2
  - 部長：基本給 \* 3



## 演習2 - 給料はいくら？

---

- 実行例

```
$ ruby shacho2.rb Tanto 100
```

担当が普通に起立しました。

給料は 100 円です。

```
$ ruby shacho2.rb Shunin 100
```

主任がすばやく立ちました。

給料は 200 円です。

## 演習2 - 給料はいくら？

---

- shain2.rb の Shain クラスに、基本給から給料を計算するメソッドを追加します。

```
class Shain
  def standup
  end

  def kyuryo(kihonkyu)
  end
end
```

## 演習2 - 給料はいくら？

---

- Tanto, Shunin, Bucho クラスの kyuryo メソッドを定義しましょう。

```
class Tanto < Shain
  def kyuryo(kihonkyu)
    return ...
  end
end
```

## 演習2 - 給料はいくら？

---

- shain2.rb (一部)

```
class Tanto < Shain
  def kyuryo(kihonkyu)
    return kihonkyu
  end
end
```

```
class Shunin < Shain
  def kyuryo(kihonkyu)
    return kihonkyu * 2
  end
end
```

## 演習3 - 取締役を追加

---

- shain3.rb に取締役を追加しましょう
  - 取締役はふんぞりかえって立ちました。
  - 取締役の給料は「基本給 \* 4」です。
- 実行例

```
$ ruby shacho3.rb Torishimariyaku 100
```

取締役はふんぞりかえって立ちました。  
給料は 400 円です。

## 演習3 - 取締役を追加

---

- shain3.rb (一部)

```
class Torishimariyaku < Shain
  def standup
    puts("取締役がふんぞりかえって立ちました。")
  end

  def kyuryo(kihonkyu)
    return kihonkyu * 4
  end
end
```

## 演習4 - ボーナスはいくら？

---

- 基本給をセットするメソッド `kihonkyu=` を定義しましょう
- ボーナスを返すメソッド `bonus` を定義しましょう。
- ボーナスは社員だれでも「給料 \* 2」です。

## 演習4 - ボーナスはいくら？

---

- 実行例

```
$ ruby shacho4.rb Tanto 100
```

担当が普通に起立しました。

給料は 100 円です。

ボーナスは 400 円です。

```
$ ruby shacho4.rb Shunin 100
```

主任がすばやく立ちました。

給料は 200 円です。

ボーナスは 400 円です。



## 演習4 - ボーナスはいくら？

---

- shain4.rb (一部)

```
class Shain
  def kihonkyu=(kihonkyu)
    @kihonkyu = kihonkyu
  end
  def kyuryo
  end
  def bonus
    return @kihonkyu * 4
  end
end
```

## 演習4 - ボーナスはいくら？

---

- shain4.rb (一部)

```
class Tanto < Shain
  def kyuryo
    return @kihonkyu
  end
end
```

```
class Shunin < Shain
  def kyuryo
    return @kihonkyu * 2
  end
end
```

## まとめ

---

- オブジェクト
  - 実際にある「もの」や「概念」
- クラス
  - オブジェクト共通の性質を持った「くくり」
- 継承
  - いろいろなクラス定義の共通化
- カプセル化
  - データ構造が変わっても、仕事の頼み方は同じ
- ポリモルフィズム
  - 仕事を頼まれる側の種類が増えても、仕事の頼み方は同じ

## 参考文献

---

- オブジェクト脳のつくり方
  - <http://www.seshop.com/detail.asp?pid=4115>
- プログラミング Ruby 第二版 言語編
  - <http://ssl.ohmsha.co.jp/cgi-bin/menu.cgi?ISBN=4-274-06642-8>

## 今後の情報源

---

- 公式 Web サイト
  - <http://www.ruby-lang.org/>
- リファレンスマニュアル
  - <http://www.ruby-lang.org/ja/man/>
- 日本 Ruby の会
  - <http://jp.rubyist.net/>
- Rubyist Magazine
  - <http://jp.rubyist.net/magazine/>
- okkez's weblog
  - <http://typo.okkez.net/>