pi-17. Javaにおけるオブジェクト指向設計とUMLクラス図

（Java マスター講座：基礎から応用まで）

URL: https://www.kkaneko.jp/pro/pi/index.html

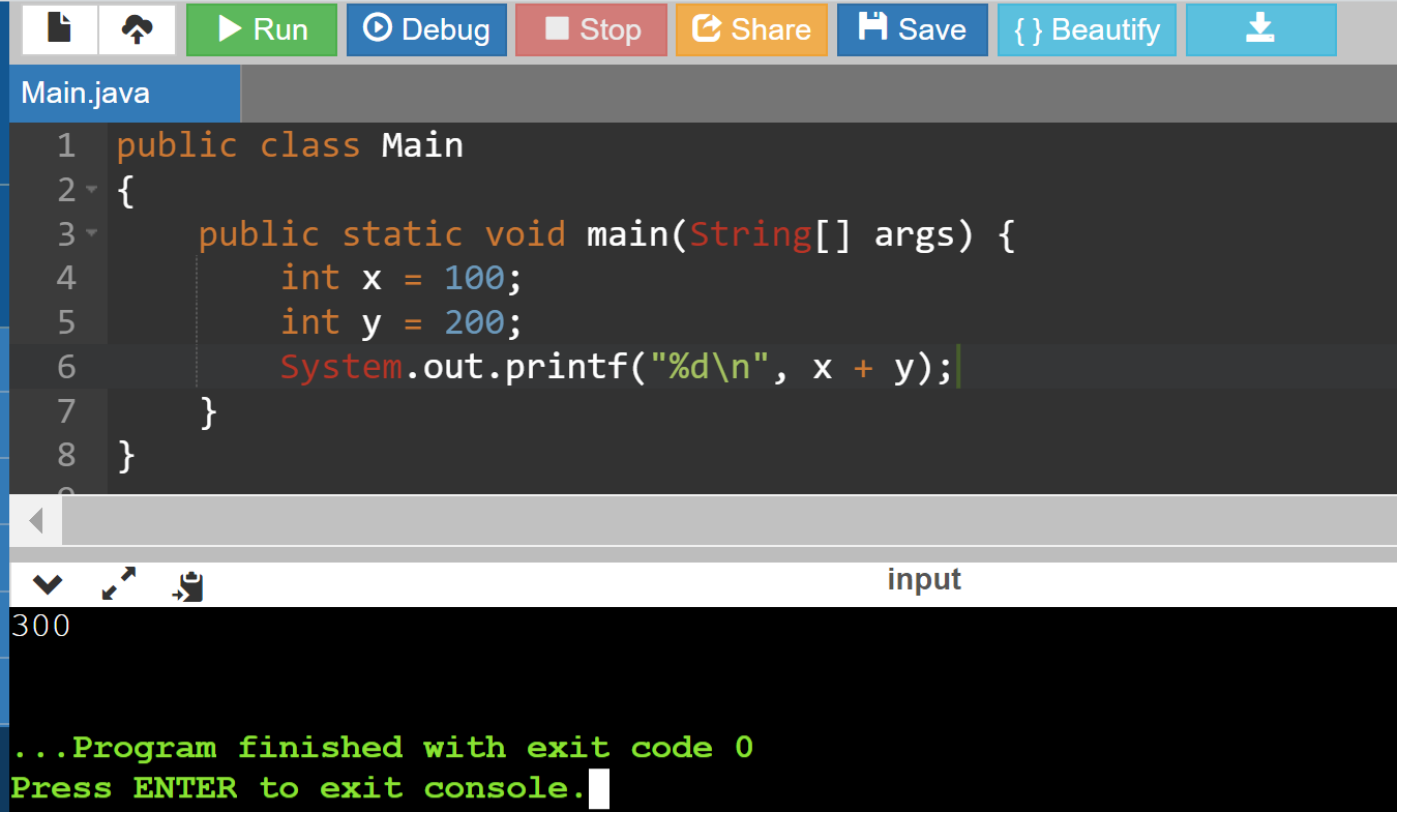
金子邦彦

CC BY NC SA

Javaにおけるクラス定義・クラス階層・継承の概念、およびUMLクラス図による設計と実装

【学習内容の構成】

- **クラス定義と継承**：属性・コンストラクタ・メソッドの定義、スーパークラスからサブクラスへの属性とメソッドの継承
- **UMLクラス図**：クラスの属性・操作・相互関係（親子関係、関連）を表現する設計図の記法
- **UMLからJavaへの実装**：クラス図をもとにしたJavaプログラムの作成
- 前提：Javaの基本データ型、基本的なプログラミング経験
- 意義：オブジェクト指向設計の理解、設計図とコードの対応関係の習得



Java などのプログラミング言語の体験，演習ができる**オンラインサービス**

GDB online

http://www.pythontutor.com/

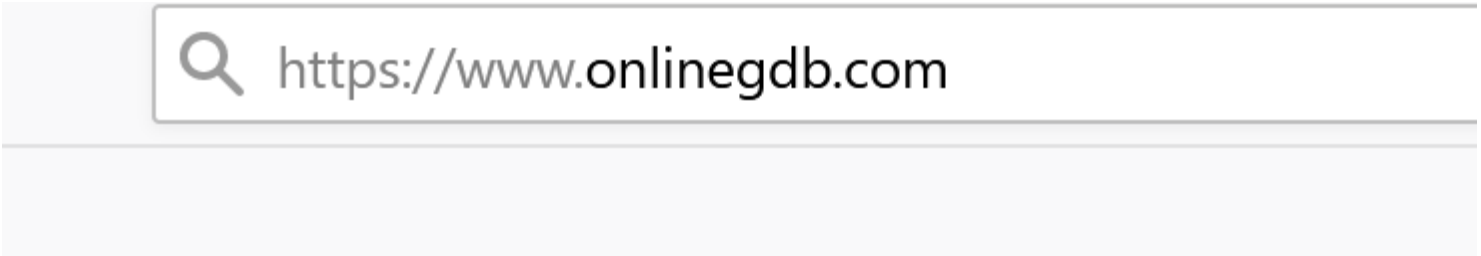
オンラインなので、「秘密にしたいプログラム」を扱うには十分な注意が必要

GDB online で Java を動かす手順

① ウェブブラウザを起動する

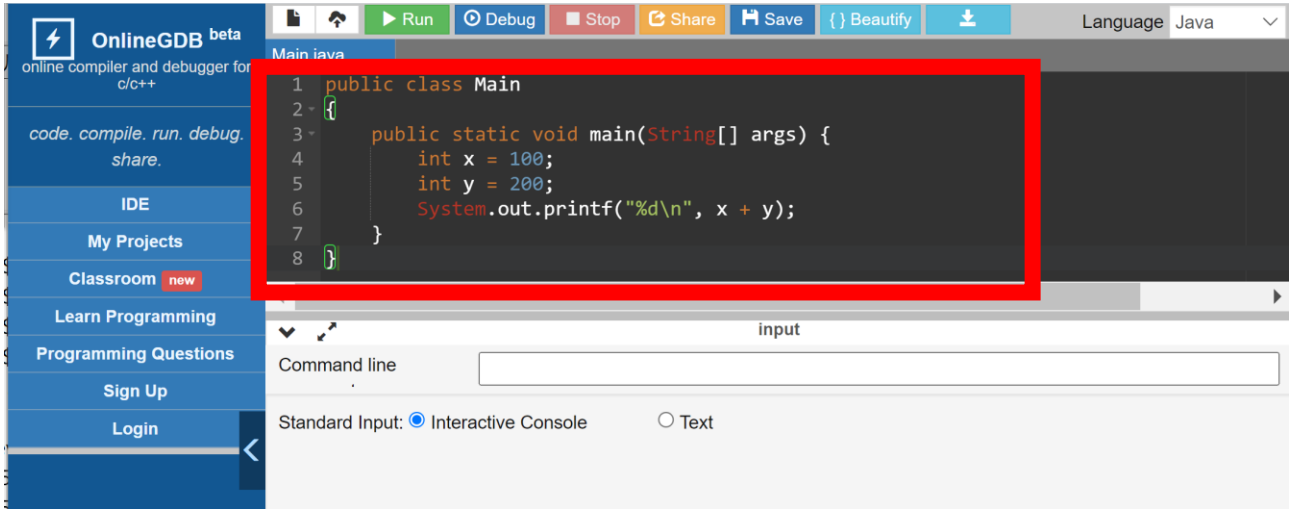
② 次の URL を開く

https://www.onlinegdb.com/



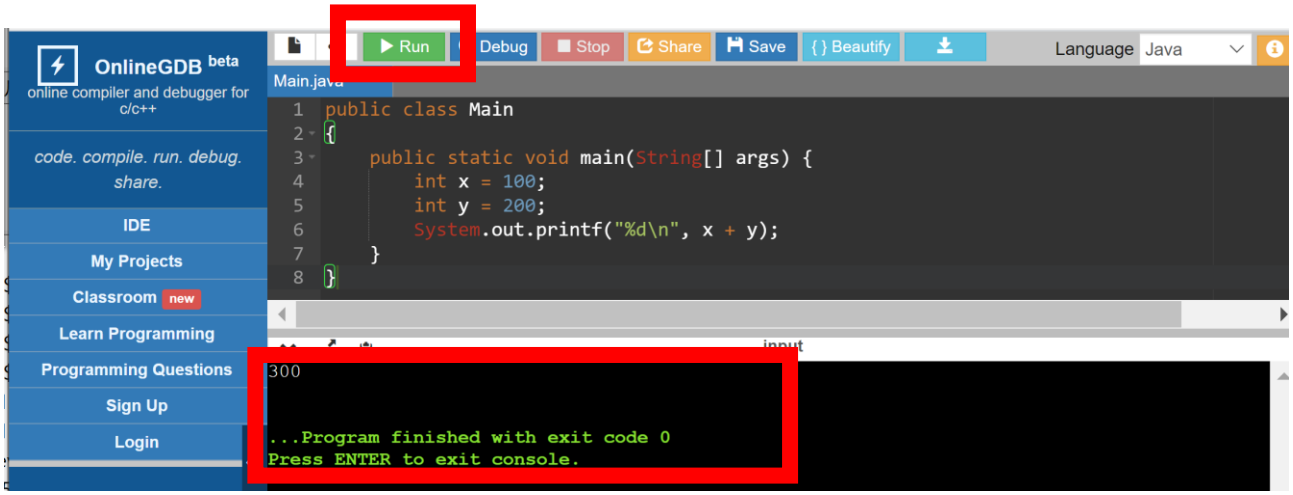
③「Language」のところで，「Java」を選ぶ

④ ソースコードを入れる



⑤ 実行．実行結果を確認

「**Run**」をクリック．



17-1. クラス定義，クラス階層，継承

Java のデータの種類

- **基本データ**

データの種類	**基本データ型**	サイズ
整数	**byte**	8 bit
	short	16 bit
	int	32 bit
	long	64 bit
浮動小数	**float**	32 bit
	double	64 bit
文字	**char**	16 bit
true/false	**boolean**	

- **基本データ**の**配列**
- **クラス**に属する**オブジェクト**: **String** クラスなど多種

クラス定義

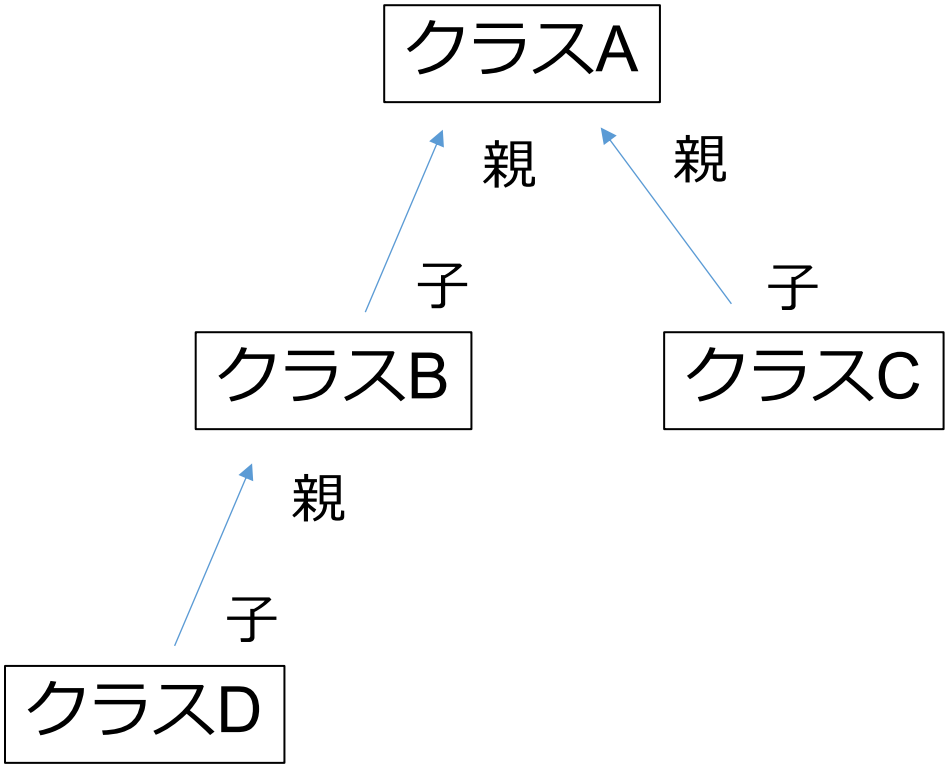
クラス定義の中には，**属性**の定義（**属性名**と**データ型**），**コンストラクタ**の定義，その他**メソッド**の定義を含める.

```
1  class C {
2    int qty;
3    double weight;
4    String name;
5    public C(int qty, double weight, String name) {
6        this.qty = qty;
7        this.weight = weight;
8        this.name = name;
9    }
10 }
```

オブジェクトの生成を行う**メソッド**のことを**コンストラクタ**という

クラス階層

クラス階層とは，**複数のクラスが親子関係をなす**こと



継承

- **継承**とは，**スーパークラス**の**属性**と**メソッド**を**サブクラス**が**受け継ぐ**こと

クラスの類似性

- 類似した２つの**クラス**

Ball

属性

x

y

color

メソッド

move

reset

x, y, color は**同じ**

r の有り無しが違う

メソッドの**名前も中身も全く同じ**とする

Circle

属性

x

y

color

r

move

reset

クラスの親子関係

```
class Ball {
  double x;
  double y;
  String color;
  public Ball(double x, double y,
String color) {
    this.x = x;
    this.y = y;
    this.color = color;
  }
  public void move(double xx, double
yy) {
    this.x = this.x + xx;
    this.y = this.y + yy;
  }
  public void reset() {
    this.x = x;
    this.y = y;
  }
}
```

クラス名 **Ball**

属性 **x, y, color**

メソッド **move, reset**

```
class Circle extends Ball {
  double r;
  public Circle(double x, double y, String color,
double r) {
    super(x, y, color);
    this.r = r;
  }
}
```

クラス名 **Circle**

属性 **x, y, color, r**

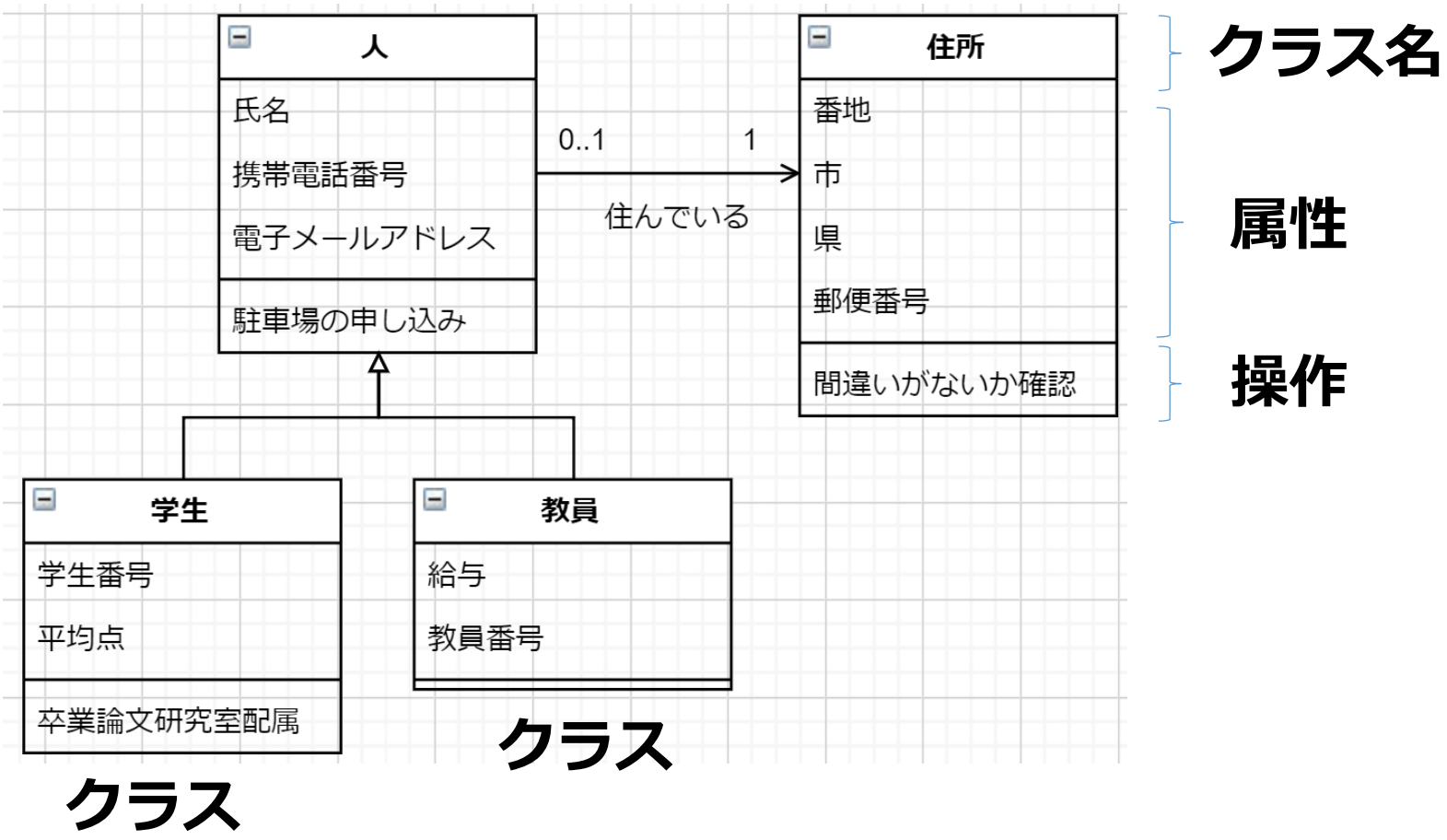
メソッド **move, reset**

クラス **Circle** は，**親クラス**であるクラス **Ball** の**属性**と**メソッド**を**継承**する.

17-2. UML のクラス図

UML のクラス図

- プログラム（ソースコード）に関する**設計図の書き方の１つ**
- **クラス**と**相互関係**，**クラス**の属性，**クラス**の操作を書く



UML

- **UML は，Unified Modeling Language のこと**
- モデリング（構造や，振る舞いや，インタラクション）の**書き方（表現）の方法を統一**すること

問いかけ

- **クラス**の**相互関係**には何があるのでしょうか？

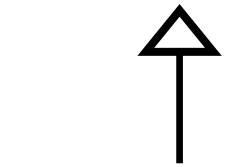
親クラス・子クラス

関連: ある**クラス**が，別の
クラスの**振る舞いや属性を呼び出す**

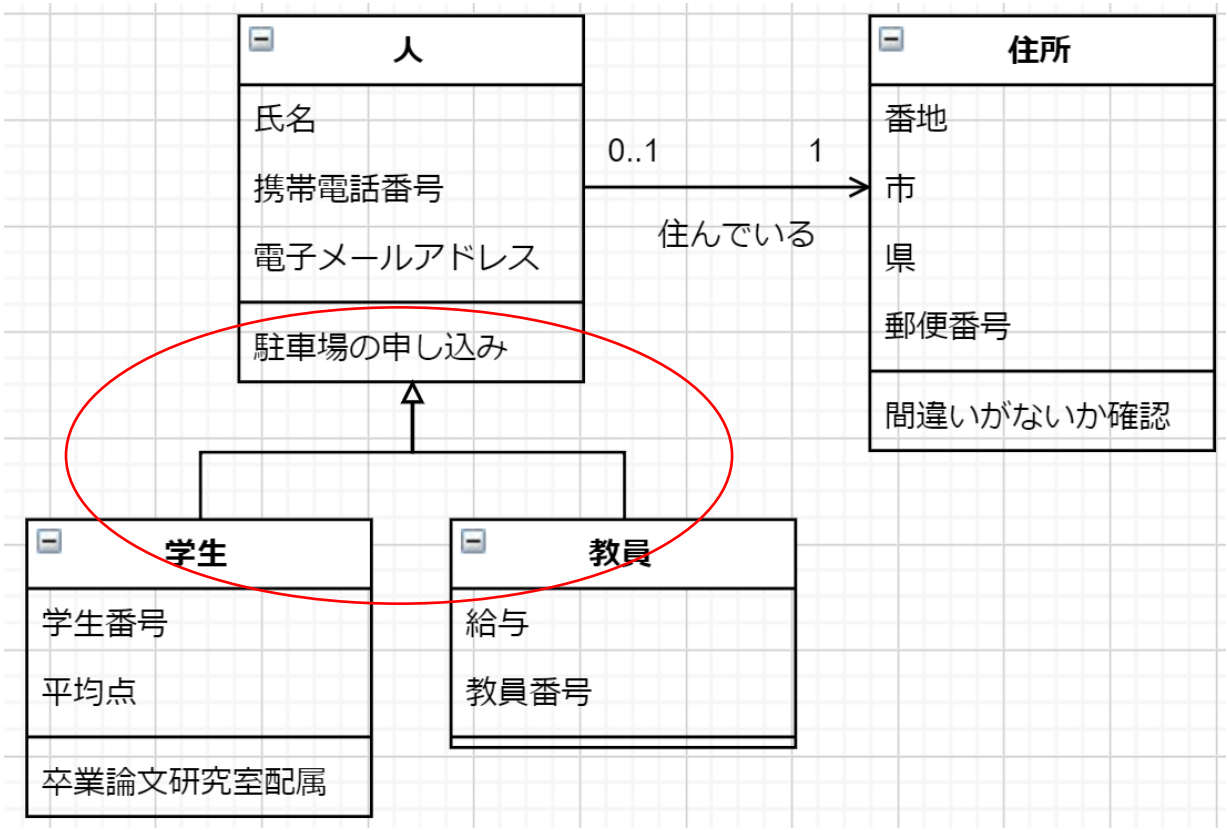
親クラス・子クラス

親クラス

三角の矢印



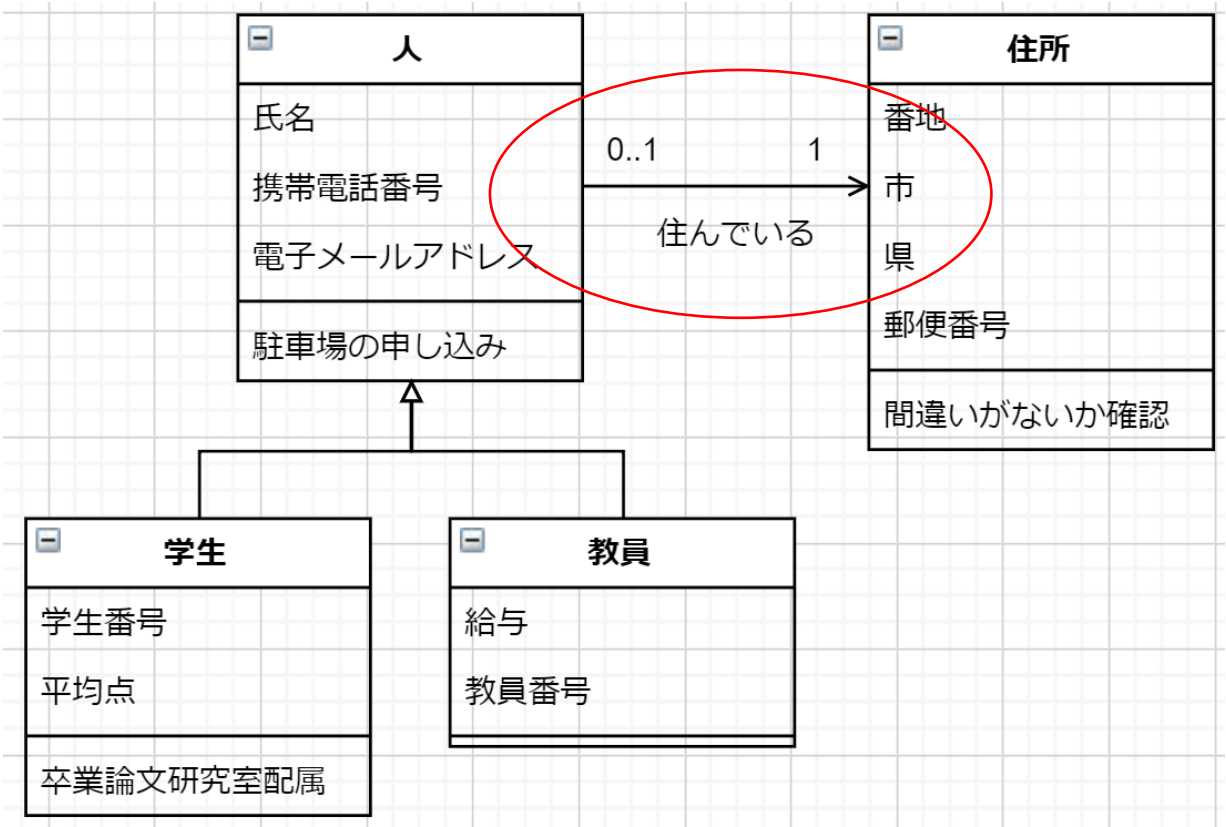
子クラス



関連

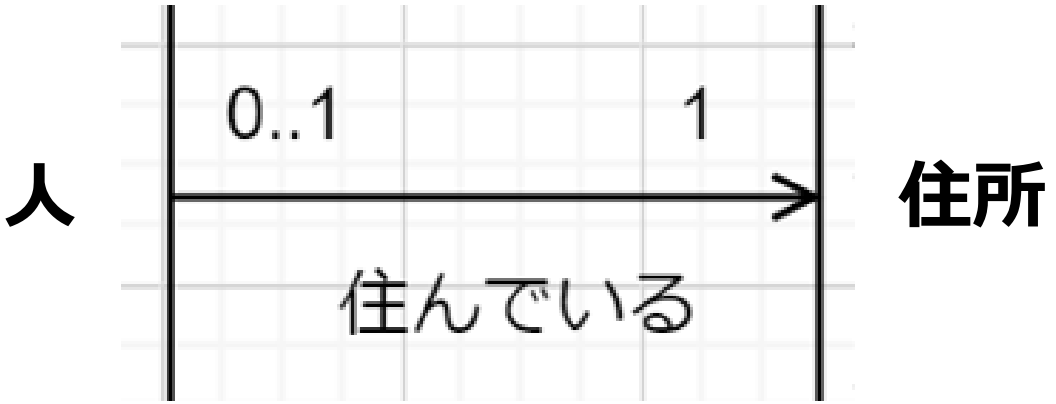
「人」についてのデータ処理をするとき，
「住所」の属性や振る舞いを呼び出す場合がある

- 「住んでいる」：呼び出す理由
- 「0..1」や「1」：**関連**の**多重度**



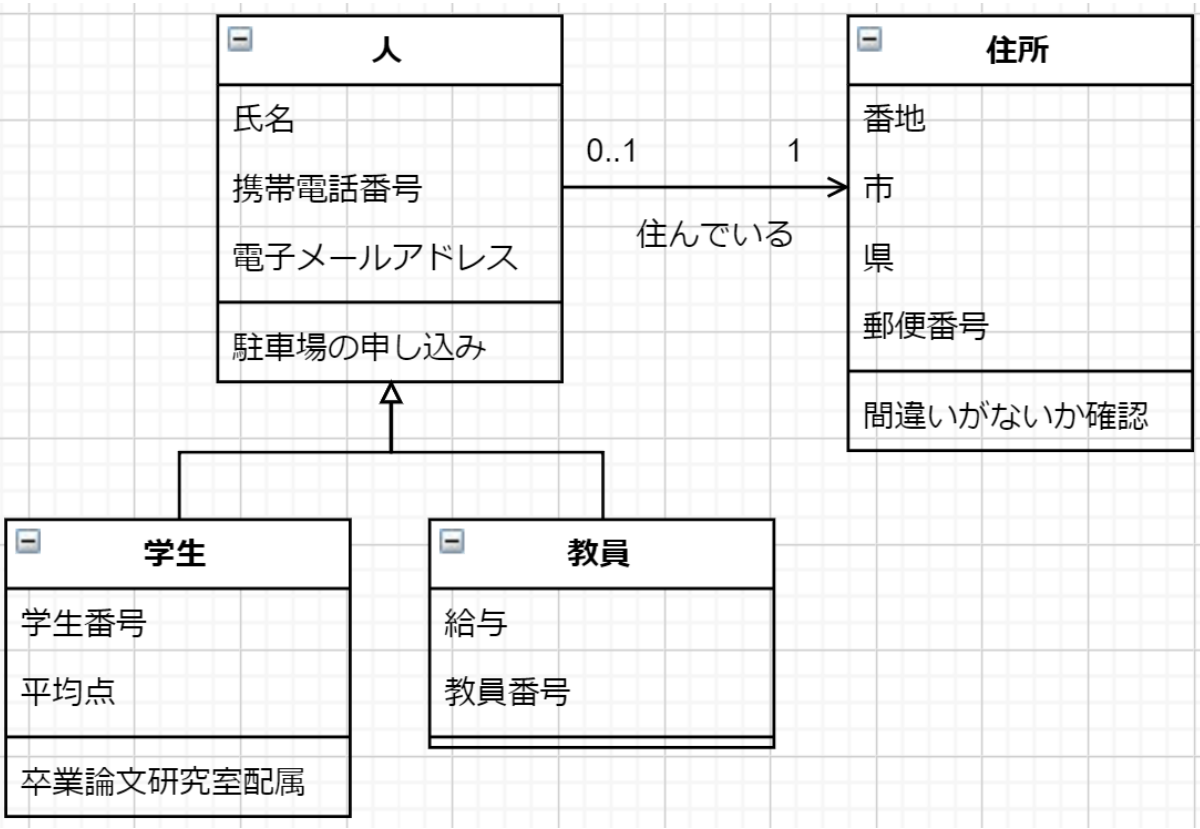
関連の多重度

指定方法	意味
0..1	0または1
1	必ず1
0..*	0以上
1..*	1以上
n	必ず*n*(ただし*n* > 1)
0..n	0以上*n*以下(ただし*n* > 1)
1..n	1以上*n*以下(ただし*n* > 1)



- １人の「人」には，住所は１つ
- 住所は
  誰もいなくなっても（引っ越しなど）
  記録は残す．同居で複数人いても記録は
  まとめない

クラス



クラス名

属性

操作

・クラスがどういう役割を担うか
・メソッド名とタイプを 「move: void」のように書くことも

UMLの特徴

- UML では，**書き方（表現）の方法を統一**するので，「人によって書き方の方法が**違う**」，「使用するプログラミング言語によって書き方の方法が**違う**」ということが**ない**
- クラス，パッケージ，オブジェクト，アクティビティ，ユースケース，ステートチャート，シーケンス，コラボレーションの書き方（表現）の書き方が統一される

※ この授業では，UMLのクラス図の演習を実施

17-3. UML のクラス図に関する実演・演習

演習

資料：**25 ～ 26**

【トピックス】

- **UMLのクラス図**

オンラインのサイト www.draw.io

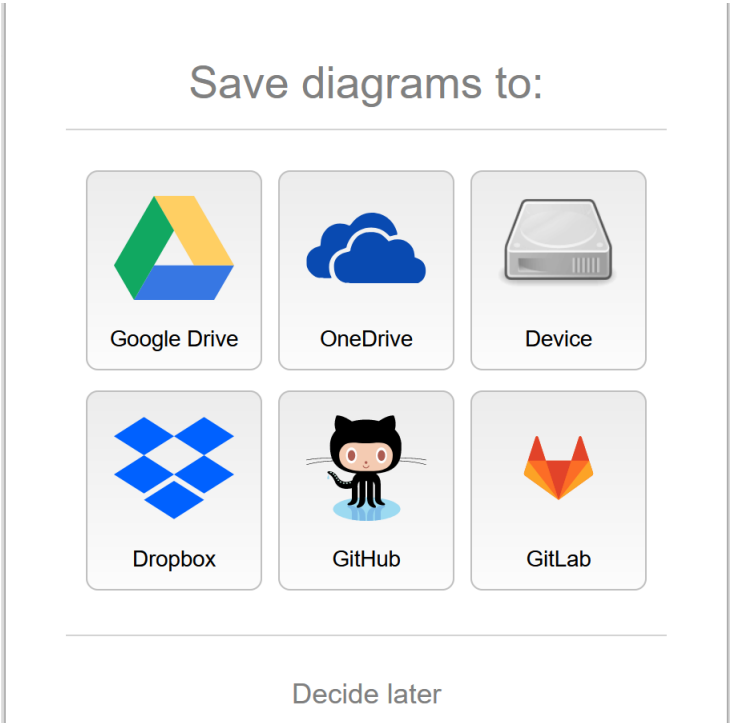
- https://www.draw.io/
- **チャート**、**設計図**等を作成できる**オンラインのサイト**
- さまざまな**シェイプ (Shape)** を**組み合わせ**て、チャート、設計図を作成
- 画像（PNG, JPEG など）、PDFファイル形式、XML形式での**エクスポート機能**もある

起動までの手順

- Webブラウザで開く

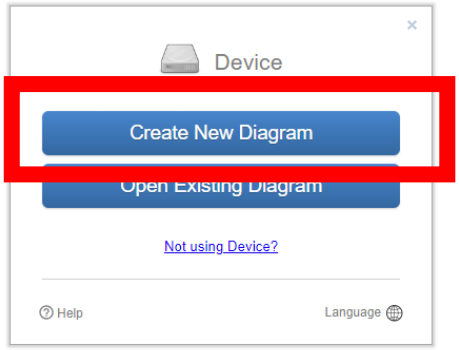
https://www.draw.io

- **保存先**が**設定**できる．保存する必要がないので「**Decide latet**」を選ぶ

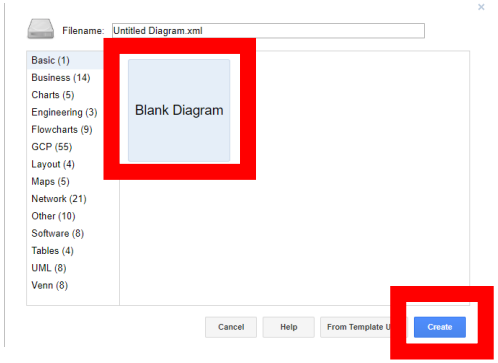


次のような画面が出ることがある（**初めて使用するときなど**）

- **新規作成**したいので「**Create New Diagram**」を選ぶ



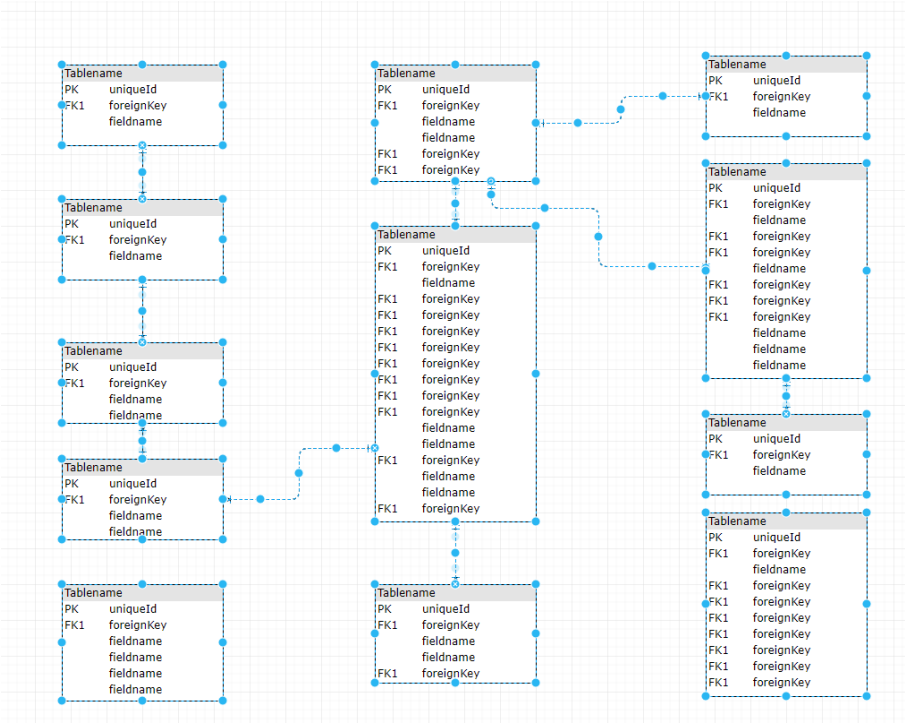
- **種類**は「**Brank Diagram**」を選び、「**Create**」をクリック



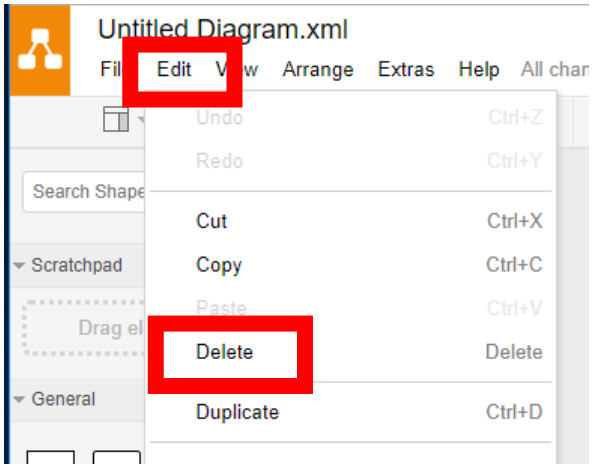
選択した
シェイプの
プロパティ

シェイプの種類

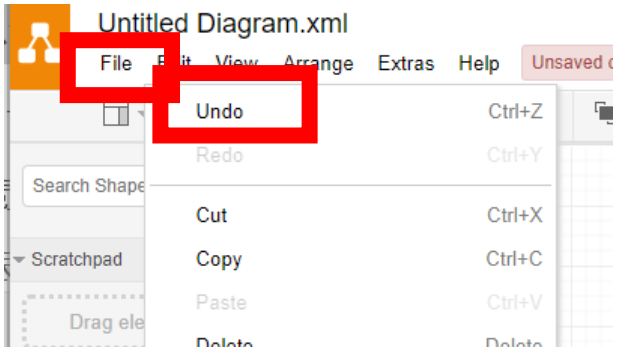
全選択：CTRL + A（同時押し）
※ CTRL は「コントロールキー」



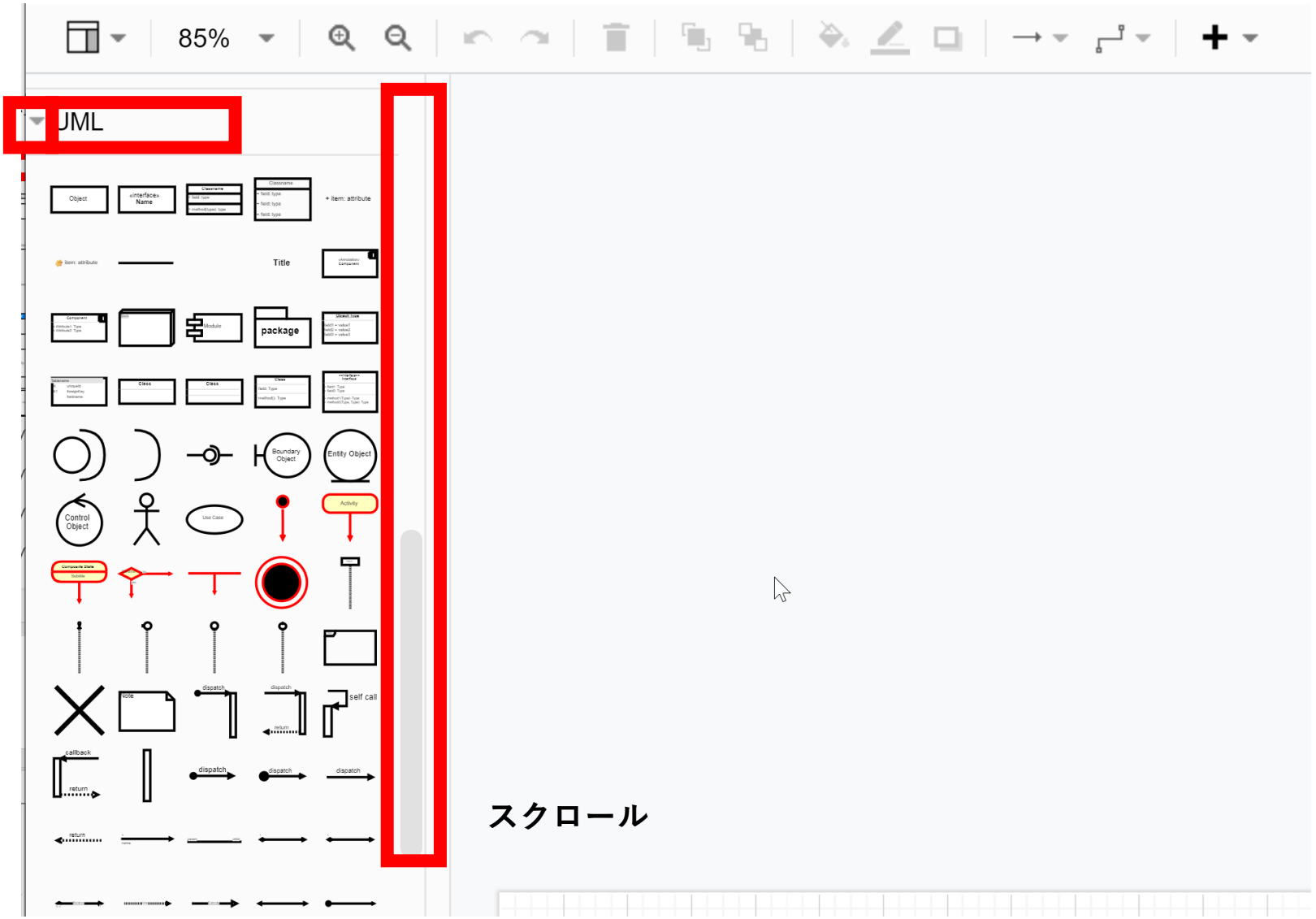
削除：メニューの「**Edit**」の「**Delete**」



アンドゥ：メニューの「Edit」の「Undo」



UML図のためのシェイプは「UML」の下

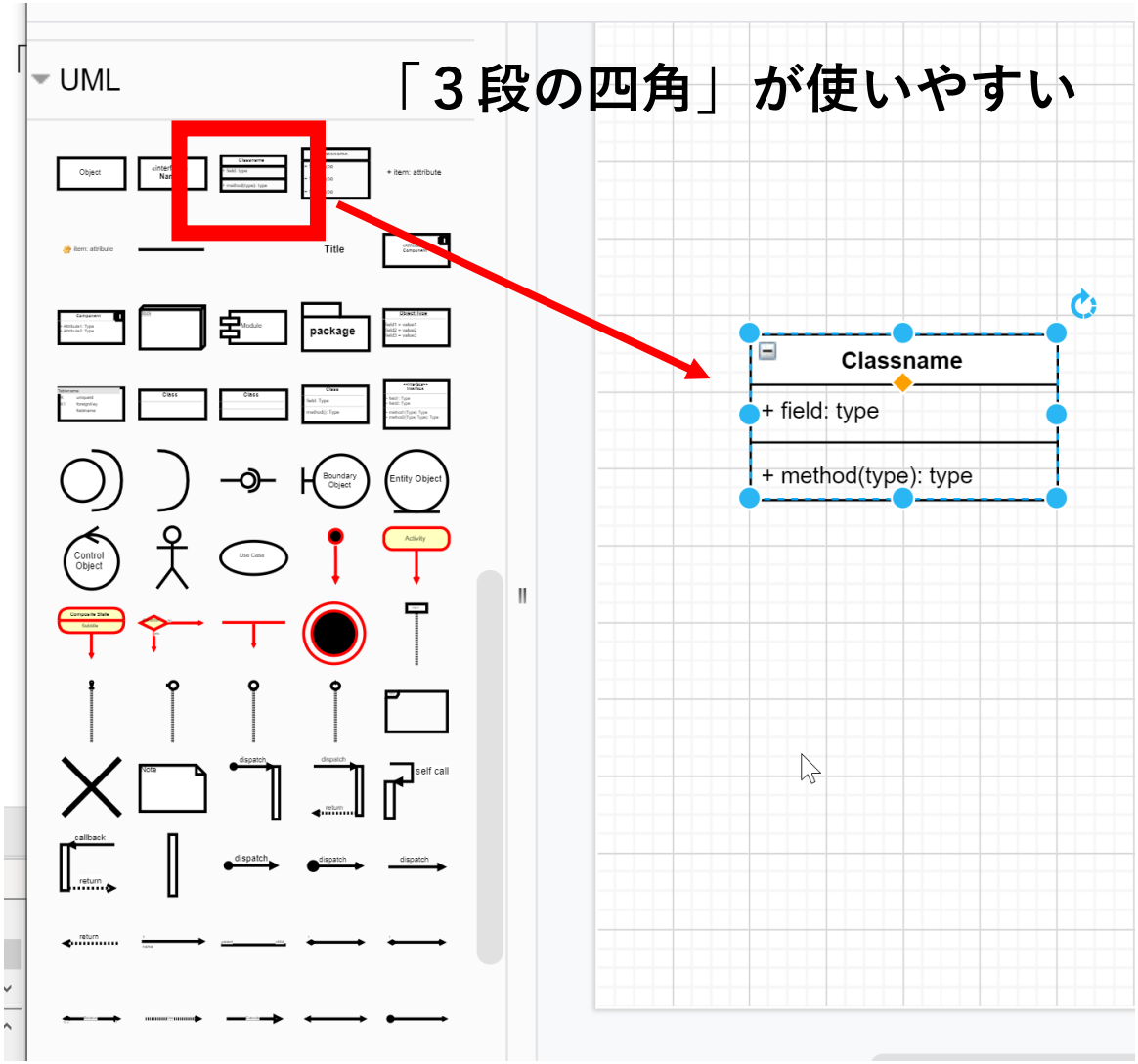


展開ボタン

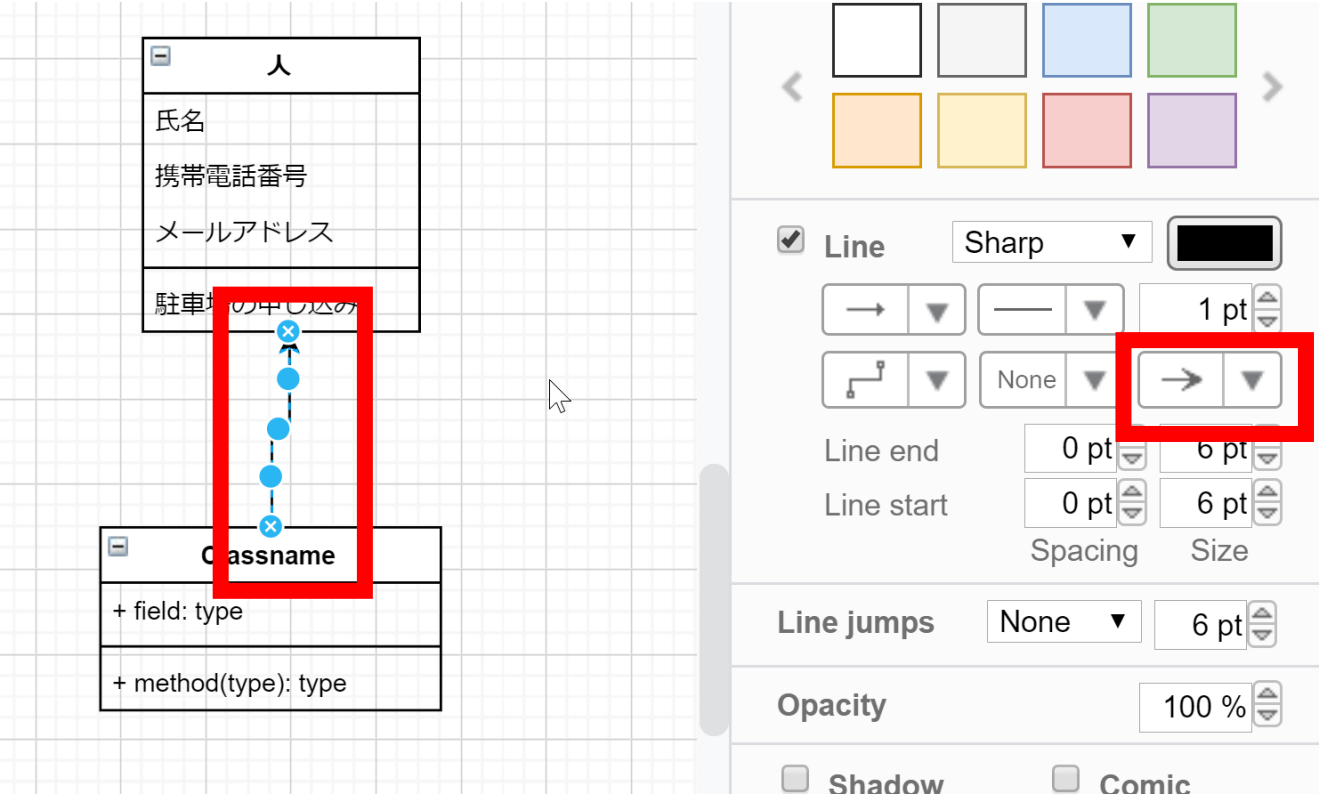
スクロール

UML図のためのシェイプは「UML」の下

クラス

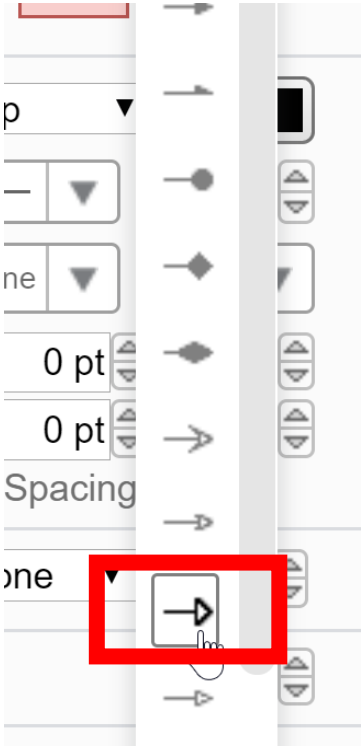


親クラス・子クラス



マウスで線を引く

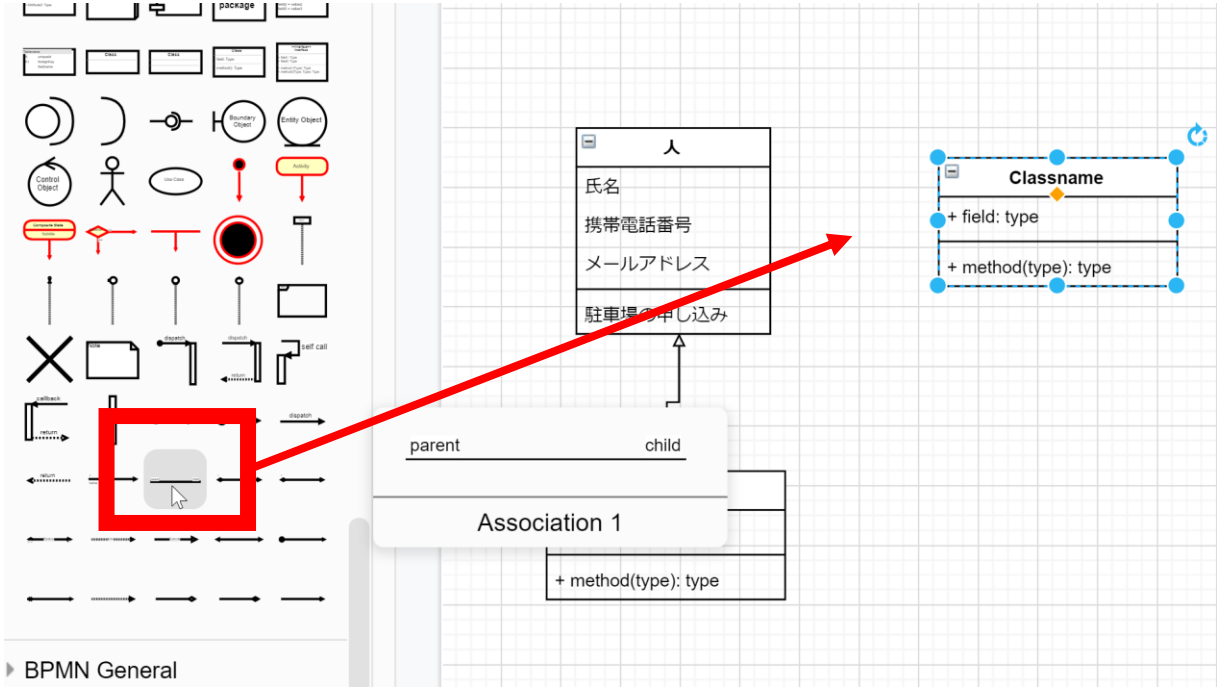
種類を選ぶ



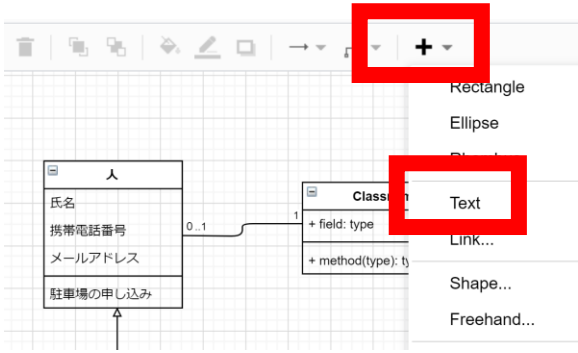
UML図のためのシェイプは「UML」の下

関連

「Association 1」が使いやすい



「0..1」や「1」は
キーボードで入れる

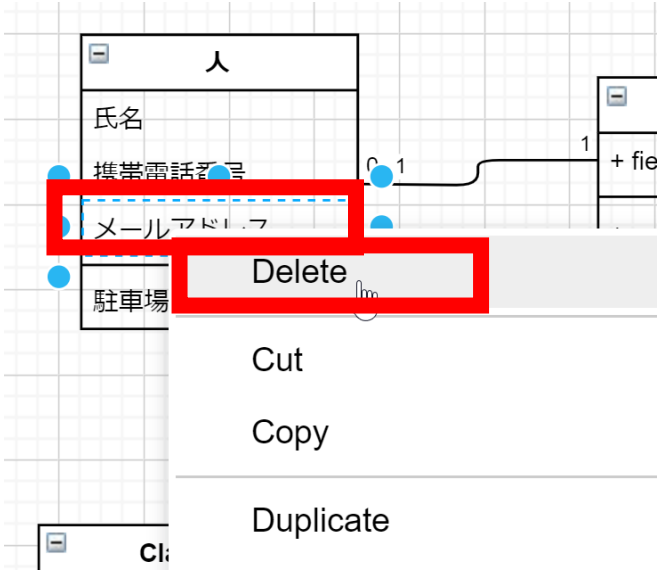


「+」，「Text」と操作し
「住んでいる」は，
キーボードで入れる

属性を減らすときは、属性を選び

右クリックメニューで

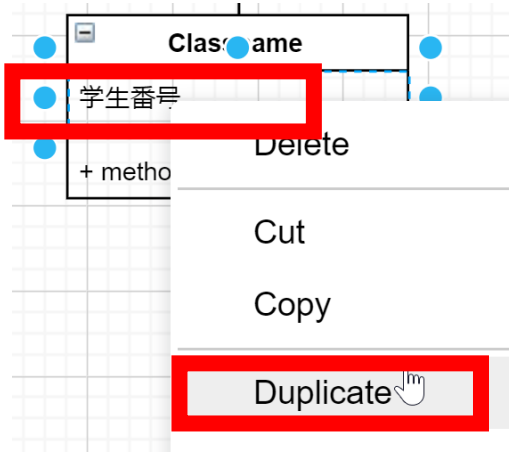
「**Delete**」

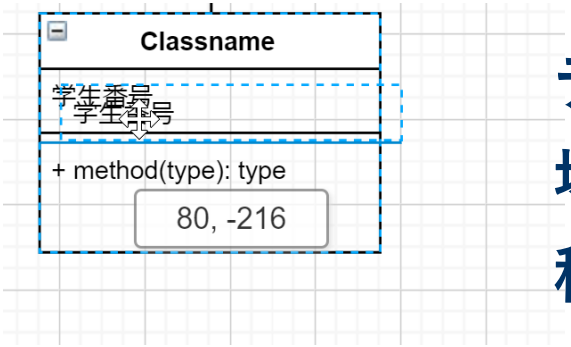


属性を増やすときは、属性を選び

右クリックメニューで

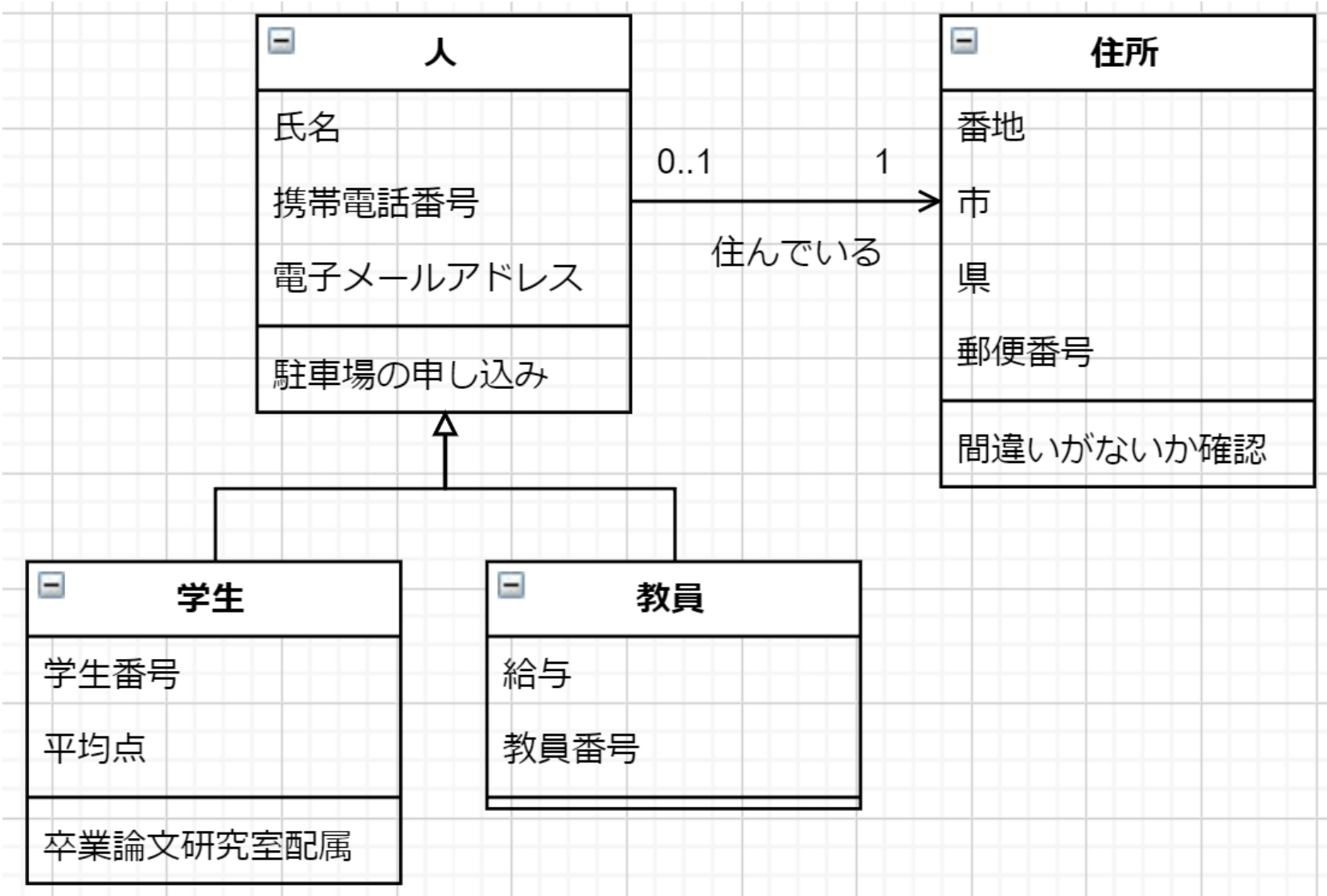
「**Duplicate**」





テキストの**場所**は，マウスで**移動できる**

次のクラス図を作成しなさい



17-4. UML のクラス図からの Java プログラム作成

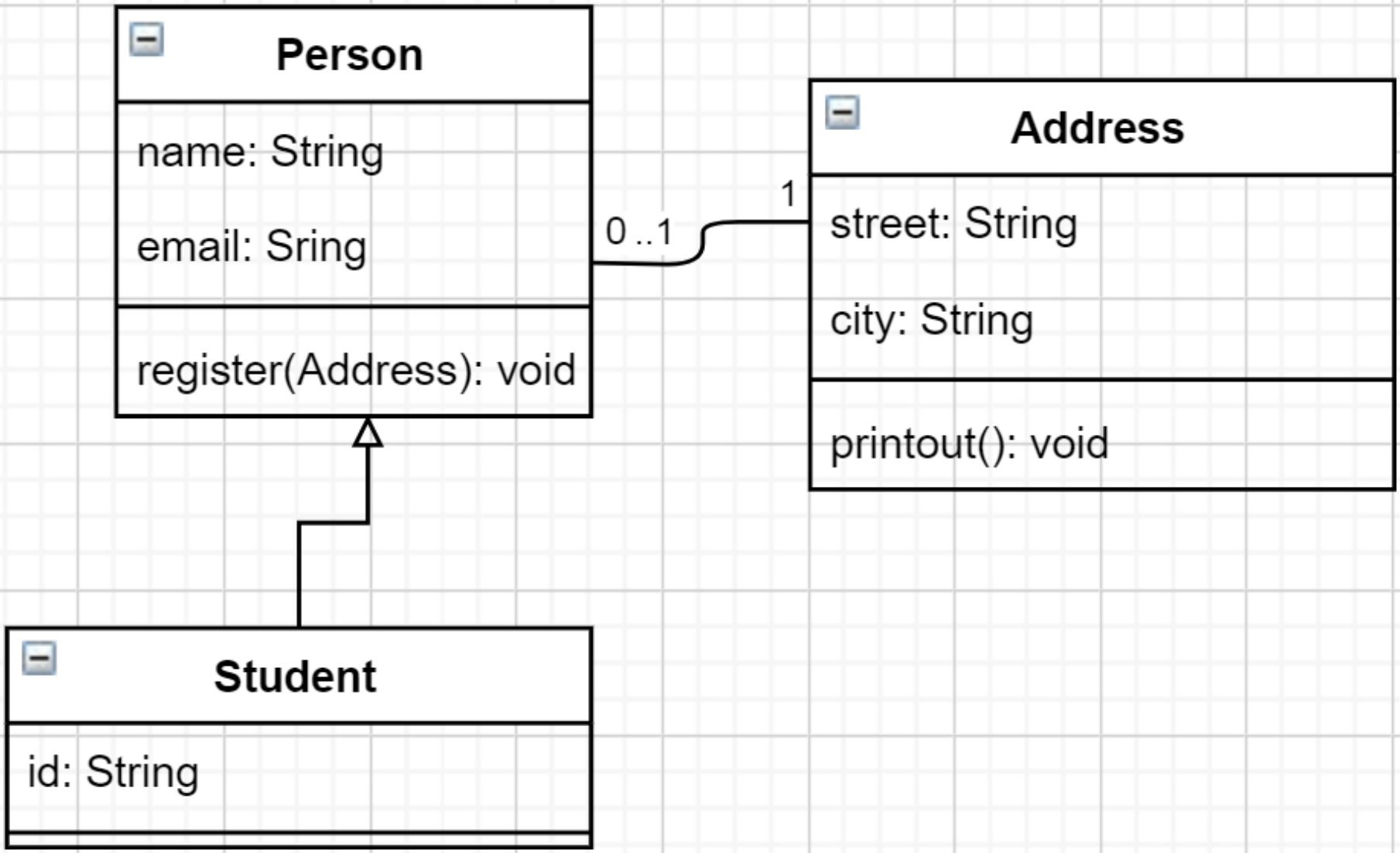
演習

資料：**39 ～ 43**

【トピックス】

- **UMLのクラス図**
- **プログラム作成**

次のクラス図の Java プログラムを演習



① GDB online の**エディタ**で，次のプログラムを入れる

Address クラス定義

```
1  class Address {
2      String street;
3      String city;
4      public Address(String street, String city) {
5          this.street = street;
6          this.city = city;
7      }
8      void printout() {
9          System.out.printf("%s %s\n", this.city, this. street);
10     }
11 }
```

② **続けて**，次のように入れる

Person クラス定義

```
12 class Person {
13     String name;
14     String email;
15     Address address;
16     public Person(String name, String email) {
17         this.name = name;
18         this.email = email;
19     }
20     public void register(Address a) {
21         this.address = a;
22     }
23 }
```

③ **続けて**，次のように入れる

Student クラス定義

```
24 class Student extends Person {
25     String id;
26     public Student(String name, String email, String id) {
27         super(name, email);
28         this.id = id;
29     }
30 }
```

④ **続けて**，次のように入れ，実行し，実行結果を確認する

```
31  public class Main
32  {
33      public static void main(String[] args) {
34          Student s = new Student("kaneko", "kaneko@hoge.com", "001");
35          Address a = new Address("1-2-3", "sanzou");
36          a.printout();
37          s.register(a);
38      }
39  }
40
```

input

```
sanzou 1-2-3


...Program finished with exit code 0
Press ENTER to exit console.
```

関連ページ

- **Java プログラミング入門**

GDB online を使用

https://www.kkaneko.jp/pro/ji/index.html

- **Java の基本**

Java Tutor, GDB online を使用

https://www.kkaneko.jp/pro/pi/index.html

- **Java プログラム例**

https://www.kkaneko.jp/pro/java/index.html