

プログラミング基礎 I

第7回 クラスと参照

九州産業大学 理工学部

pk@is.kyusan-u.ac.jp

シラバスの講義計画を一部変更

- クラスの内容を第7回にスライド

講義回	内容
1	変数、代入、演算
2	条件分岐
3	繰り返し
4	条件分岐と繰り返し
5	配列と参照 (1)
6	配列と参照 (2)
7	メソッド基礎 (1)
8	メソッド基礎 (2)
9	メソッド応用 (1)
10	メソッド応用 (2)
11	クラス (1)
12	クラス (2)
13	総合演習
14	まとめ

講義回	内容
1	変数、代入、演算
2	条件分岐
3	繰り返し
4	条件分岐と繰り返し
5	配列と参照 (1)
6	配列と参照 (2)
7	クラス (1)
8	クラス (2)
9	メソッド基礎 (1)
10	メソッド基礎 (2)
11	メソッド応用 (1)
12	メソッド応用 (2)
13	総合演習
14	まとめ

講義計画の変更理由

- 参照型である配列とクラス型変数を続けて学んだ方が、オブジェクト指向型プログラミング言語の理解が進むとの考え

プログラムを書く上での注意

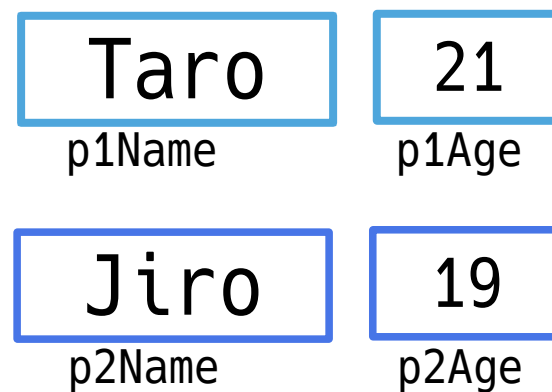
- 括弧の対応に気を付ける
 - 開き括弧と閉じ括弧の数は必ず同じになる
 - { } をブロックとして見て、区切りの位置に気を付ける
 - 特に今回は要注意！！
- インデントの自動訂正
 - プログラム全体を選択（**ctrl+A**）して、
「ソース」→「インデントの訂正」（**ctrl+I**）
で、ある程度自動でインデントを整えられる。

内容

- クラス

2人の名前と年齢を扱うプログラム (これまでの方法)

```
public class Foo {  
    public static void main(String[] args) {  
        String p1Name;  
        int p1Age;  
        String p2Name;  
        int p2Age;  
        p1Name = "Taro";  
        p1Age = 21;  
        p2Name = "Jiro";  
        p2Age = 19;  
        System.out.println(p1name+"さん("+p1Age+"歳)");  
        System.out.println(p2name+"さん("+p2Age+"歳)");  
    }  
}
```



名前の付け方で見分けてはいるが、
どこかで取り違える可能性も高くなる

クラス

```
class Person {  
    String name;  
    int age;  
}
```

これは何？

データの種類の宣言
String nameとint ageを
まとめたPersonという型

```
public class CTest01 {
```

```
    public static void main(String[] args) {
```

```
        Person p1 = new Person();
```

```
        Person p2 = new Person();
```

```
        p1.name = "Taro";
```

```
        p1.age = 21;
```

```
        p2.name = "Jiro";
```

```
        p2.age = 19;
```

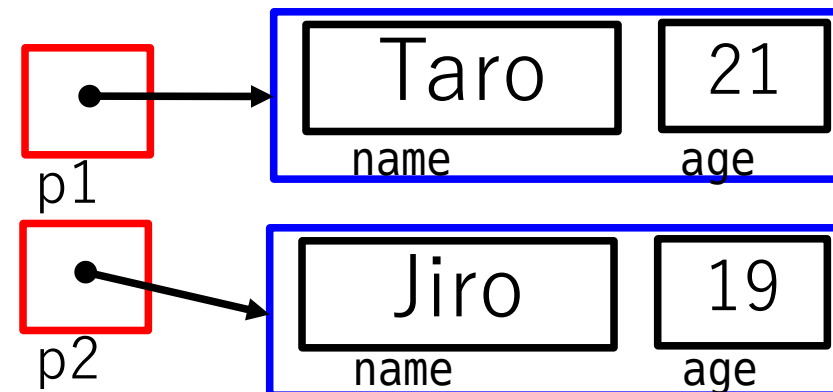
```
        System.out.println(p1.name+"さん("+p1.age+"歳)");
```

```
        System.out.println(p2.name+"さん("+p2.age+"歳)");
```

```
    }
```

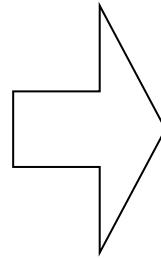
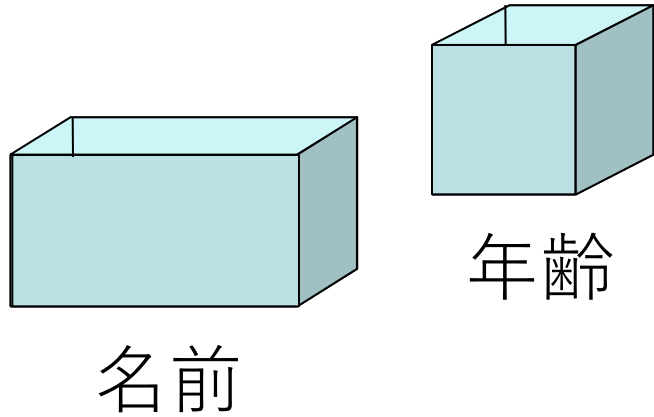
```
}
```

変数が **2** つですむ



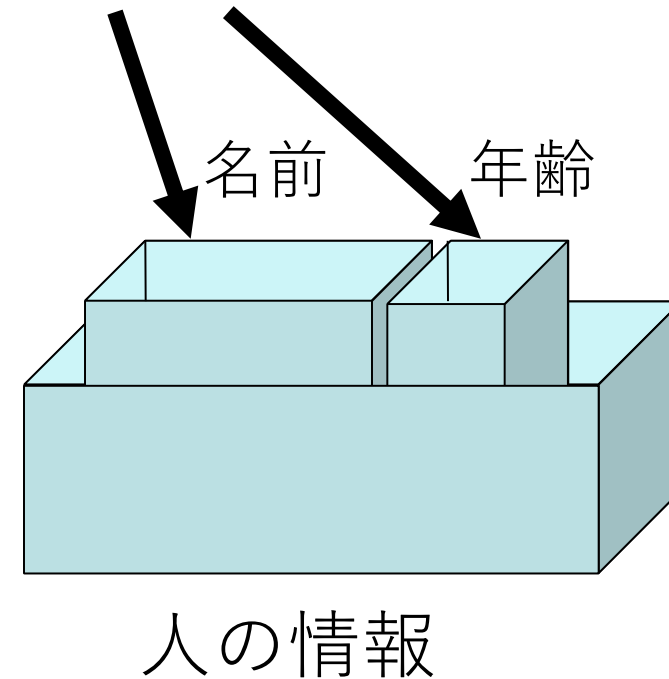
複数のデータをクラスでまとめる 教科書P108

ばらばらなデータ



フィールドのみのクラス
(レコード)

フィールド (クラスの内部の変数)



フィールド（変数）のみのクラスの宣言

```
class クラス名 {  
    型1 フィールド名1;  
    :  
    型n フィールド名n;  
}
```

まとめたい
データ

変数の宣言と
同じ書き方

例 名前と年齢をまとめて扱いたい

Personクラスの宣言 String型のnameとint型のageを持つ

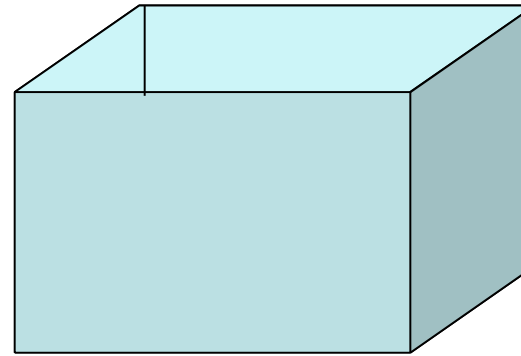
```
class Person {  
    String name;  
    int age;  
}
```

クラス型の変数の変数宣言

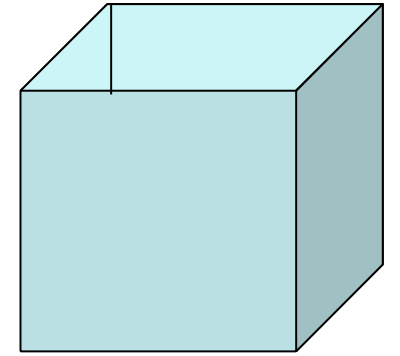
型 変数名;

String name;

int age;



String name

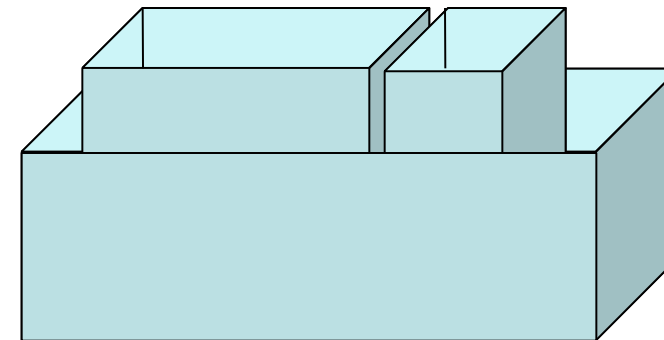


int age

Person型の変数 p1の宣言

Person p1;

String name int age



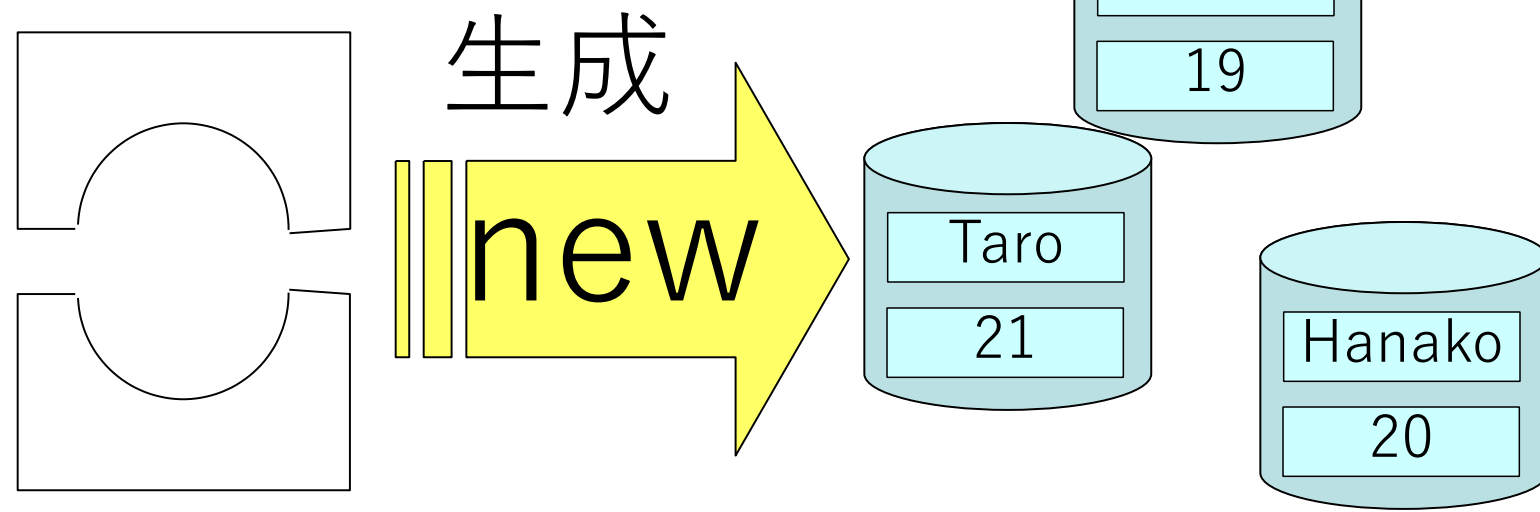
Person p1

変数宣言の書き方は同じ

データを格納するには クラスからインスタンスを生成

クラス
(型, 設計図, 工場)

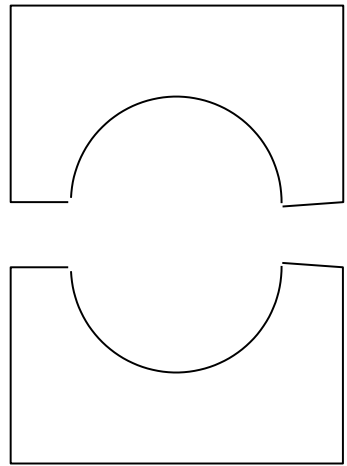
インスタンス
(オブジェクト)



クラスとオブジェクトのイメージ

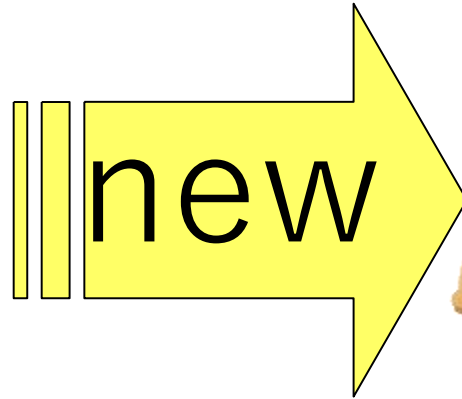
クラス
(型, 設計図, 工場)

インスタンス
(オブジェクト)



たい焼きの金型

生成



あんこ



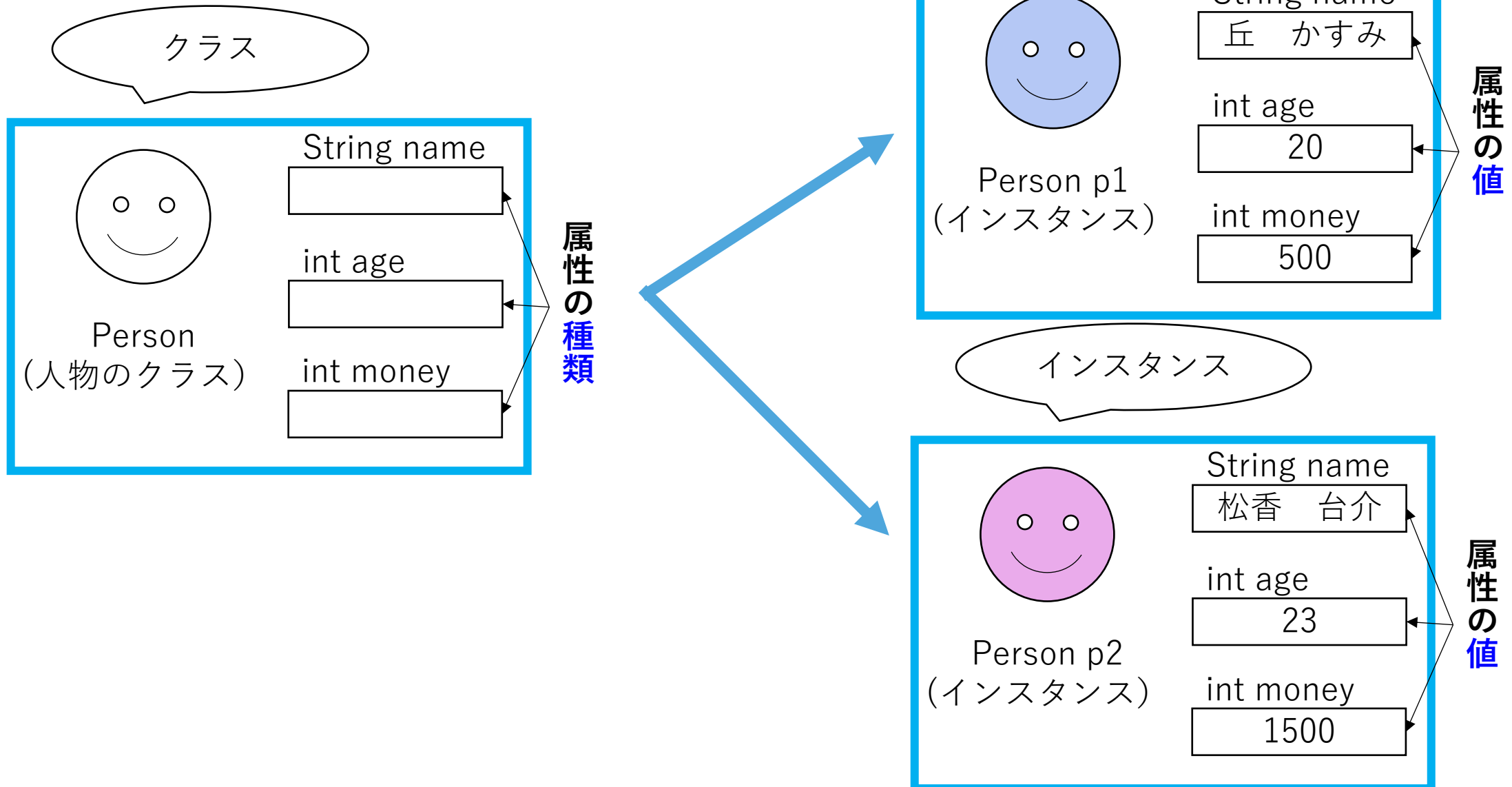
抹茶



カスタード

金型から実物のたい焼きが生成される

クラスとインスタンス



複数のデータをひとまとめにして扱うことができる

インスタンスの生成と使用

インスタンスの生成 **new クラス名 ()**

例 Personクラスの変数p1を宣言し生成

```
Person p1;           // 宣言
```

```
p1 = new Person();   // 生成
```

宣言と同時に生成

```
Person p1 = new Person();
```

クラスのフィールドへのアクセス **インスタンスの変数名.フィールド名**

例 p1のフィールドnameに"丘 かすみ"を代入

```
p1.name = "丘 かすみ";
```

例 p1のフィールドageに21を代入

```
p1.age = 21;
```

■は「の」

と読める。

- オブジェクト指向プログラミング言語
 - クラスやインスタンスなどの仕組みを備えるプログラミング言語
 - Javaはオブジェクト指向プログラミング言語
- インスタンスとオブジェクト
 - Javaでは、インスタンスとオブジェクトは同じものを指す
(観点の違いによって表現が異なる)
 - インスタンス：何らかのクラスから生成されたモノ
 - オブジェクト：記憶領域を占有するモノ

インスタンス(オブジェクト)の生成

参照型のデータをインスタンスと呼ぶ

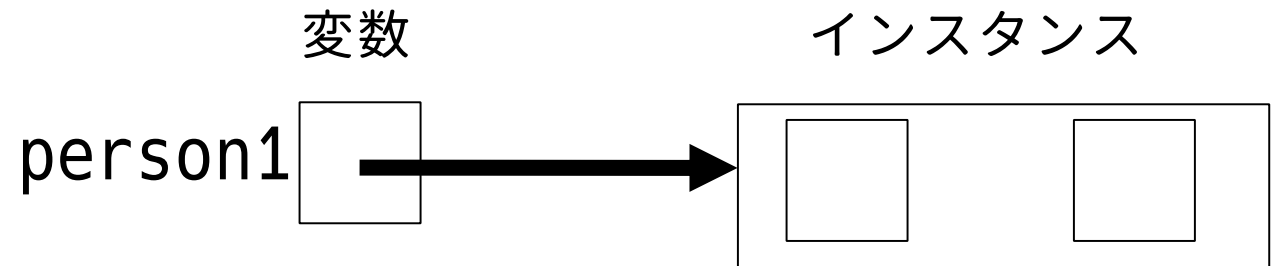
インスタンスは変数を宣言しただけでは生成されない



インスタンスは **new** 演算子 で生成

構文 変数名 = **new** クラス名();

person1 = new Person();



宣言と生成をまとめて

Person person1 = new Person();

例：人物を扱うクラス Person

- クラス名：Person
- フィールド：String型のname（名前）
int型のage（年齢）

```
class Person {  
    String name;  
    int age;  
}
```

Person クラス宣言

```
Person tarou = new Person();  
tarou.name = "理工太郎";  
tarou.age = 19;
```

Person型変数の宣言

Person型のインスタンスを生成して代入 18

例：科目情報を扱うクラスSubject

- クラス名： Subject
- フィールド：String型のname（名前）
int型の credits（単位数）
String型の className（教室名）

```
class Subject {  
    String name;  
    int credits;  
    String className;  
}
```

```
Subject math = new Subject();  
math.name = "数学";  
math.credits = 2;  
math.className = "12107";
```

例：学生情報を扱うクラスStudent

他のクラスもフィールドにすることが出来る

- クラス名： Student
- フィールド：String型のstudentNumber(学籍番号)
 Person型のperson(人物)
 Subject型のsubject1(履修科目1);
 Subject型のsubject2(履修科目2);

```
class Student {  
    String studentNumber;  
    Person person;  
    Subject subject1;  
    Subject subject2;  
}
```

```
Student student1 = new Student();  
student1.studentNumber = "25RS999";  
student1.person = tarou;  
student1.subject1 = math;  
student1.subject2 = english;
```

クラス名の命名規則

- **全て半角英数**とする。全角文字を含んではならない。
- 最初の文字は**大文字**英字とする
 - インスタンス名は小文字で開始、クラス名(型名)は大文字で開始
- 複数の単語を繋げたクラス名とする場合は、**最初の単語を大文字**とする。
(大文字キャメルケース (UpperCamelCase))
 - Class **F**oodInfo { } Class **S**tudentInfo { }

クラスを宣言する場所

一番上の行に新しいクラス宣言を記述しておくこと、
クラスの宣言場所を間違えるミスを防止しやすい

```
class 宣言するクラス名{
```

```
}
```

すでに宣言されているクラスの宣言部の上の行

```
public class Main {
```

すでに宣言されているクラス

```
    public static void main (String[] args) {
```

```
    }
```

```
}
```

クラスの中にクラスを宣言しないように注意すること

```
1 class Person {  
2     String name;  
3     int age;  
4 }  
5 class Subject {  
6     String name;  
7     int credits;  
8     String className;  
9 }
```

独自クラスを宣言する場所
(Mainクラスの上の行)

Eclipseの[クラス作成]で自動生成される最初行

```
10  
11 public class Main07p1 {  
12     public static void main(String[] args) {
```

```
14         Person tarou = new Person(); //Person型のtarou  
15         Subject subject1 = new Subject(); //Subject型のsubject1
```

クラス型変数の宣言はmainメソッドの中

```
16  
17     }  
18 }
```

```
1 class Person {  
2     String name;  
3     int age;  
4 }  
5 class Subject {
```

独自クラスを宣言する場所
(Mainクラスの上の行)

クラスの宣言以外は、
全てmainメソッド内に記述する

```
11 public class Main07pt {  
12     public static void main(String[] args) {  
13           
14         Person tarou = new Person(); //Person型のtarou  
15         Subject subject1 = new Subject(); //Subject型のsubject1  
16           
17     }  
18 }
```

クラス型変数の宣言はmainメソッドの中

注意！！！！

- クラスの中にクラスを宣言しない
- mainメソッドの中にクラスを宣言しない
- 宣言したクラスの中に別のクラスを宣言しない
- エラーは出ない（場合もある）が、正しく動作しない
 - 内部クラスというクラスの使い方もあるが、この授業では説明しない
 - まずは、基本的なクラスの宣言方法、使い方を覚えよう

クラスを宣言する

- Mainクラスの外に宣言

```
class Person {  
    String name;  
    int age;  
}
```

Personクラスを宣言（作る）ことで、
Person型の変数を扱えるようになる
（設計図がないものは作れない！）

```
public class Main07p1 {  
    public static void main(String[] args) {  
        Person tarou = new Person(); //Person型のtarou  
    }  
}
```

クラス型変数を宣言、インスタンス生成

- mainクラス内に、クラス型変数を宣言
インスタンスを生成して代入

```
class Person {  
    String name;  
    int age;  
}  
  
public class Main07p1 {  
    public static void main(String[] args) {  
        Person tarou = new Person(); //Person型のtarou  
    }  
}
```

Person型の変数には、Person型のインスタンスを生成

インスタンスのフィールドに値を代入

- Person型のインスタンスのフィールドに値を代入

```
public class Main07p1 {  
    public static void main(String[] args) {  
        Person tarou = new Person(); //Person型のtarou  
        tarou.name = "理工太郎";  
        tarou.age = 18;  
    }  
}
```

Person型のインスタンスを生成、変数名は tarou

tarou.name に "理工太郎" を代入
tarouの名前

tarou.age に 18 を代入
tarouの年齢

インスタンスのフィールドを表示

```
public class Main07p1 {  
    public static void main(String[] args) {  
        Person tarou = new Person(); //Person型のtarou  
        tarou.name = "理工太郎";  
        tarou.age = 18;  
        System.out.println(tarou.name + "(" + tarou.age + "歳)");  
    }  
}
```

tarou.name
tarouの名前

tarou.age
tarouの年齢

Subjectクラスを宣言

- Subjectクラスを宣言

```
class Person {  
    String name;  
    int age;  
}  
  
class Subject {  
    String name;  
    int credits;  
    String className;  
}  
  
public class Main07p1 {  
    public static void main(String[] args) {  
  
        Person tarou = new Person();  
        tarou.name = "理工太郎";  
        tarou.age = 18;  
    }  
}
```

クラス型変数を宣言、インスタンス生成

- インスタンスのフィールドに値を代入

```
System.out.println(tarou.name + "(" + tarou.age + "歳)");  
↵
```

```
Subject math = new Subject();  
math.name = "数学";  
math.credits = 2;  
math.className = "12107";  
↵
```

```
Subject english = new Subject();  
english.name = "英語";  
english.credits = 1;  
english.className = "S301";  
↵
```

インスタンスのフィールドを表示

```
System.out.println(math.name + "(" + math.credits + "単位):" + math.className + "教室");  
System.out.println(english.name + "(" + english.credits + "単位):" + english.className + "教室");
```

- 一連の流れを覚えて身につけよう

クラスのフィールドに別クラスを定義

- Studentクラスを宣言、
Person型、Subject型のフィールドを持つ

```
5 class Subject {  
6     String name;  
7     int credits;  
8     String className;  
9 }  
10 class Student {  
11     String studentNumber;  
12     Person person;  
13     Subject subject1;  
14     Subject subject2;  
15 }
```

Student型の変数を宣言して使う

```
Student student1 = new Student();  
student1.studentNumber = "25RS999";  
student1.person = tarou;  
student1.subject1 = math;  
student1.subject2 = english;
```

- student1 の person に tarou を代入
- student1 の subject1 に math を代入

すでにあるインスタンスを再利用することが出来る

インスタンスのフィールドを表示

- . で繋いでインスタンスのフィールドにアクセス

```
student1.subject2 = english;
System.out.print(student1.studentNumber + " " + student1.person.name + "の");
System.out.print(student1.subject1.name + "の教室:" + student1.subject1.className);
```

- student1.studentNumber student1の学籍番号
- student1.person.name student1の人物情報の名前
- student1.subject.name student1の履修情報の名前

長くなるが、どの情報が誰に属しているのかが分かりやすい
(メソッドを使えるようになると、短く分かりやすく書けるようになる)

よくある間違い

- クラスを宣言していない

```
1 public class Main07p1 {  
2     |   public static void main(String[] args) {  
3     |  
4     |       Person tarou = new Person();           //Person型のtarou  
5     |       tarou.name = "理工太郎";  
6     |       tarou.age = 18;  
7     |  
8     |       System.out.println(tarou.name + "(" + tarou.age + "歳)");  
9     |   }  
10  }  
|0
```

考えずにプログラムをただ書き写している時に起こりがち

存在しないクラスの変数は宣言できない

よくある間違い

- クラスの中にクラスを宣言している

```
public class Main07p1 {  
    class Person {  
        String name;  
        int age;  
    }  
}
```

クラス内にクラスを作らない！

```
public static void main(String[] args) {  
    Person tarou = new Person(); //Person型のtarou  
    tarou.name = "理工太郎";  
    tarou.age = 18;  
    System.out.println(tarou.name + "(" + tarou.age + "歳)");  
}
```

そもそも、エラーになる

よくある間違い

- インスタンスを生成していない (new していない)

```
public class Main07p1 {  
    public static void main(String[] args) {  
  
        Person tarou; //Person型のtarou  
        tarou.name = "理工太郎";  
        tarou.age = 18;  
  
        System.out.println(tarou.name + "(" + tarou.age + "歳)");  
    }  
}
```

インスタンスは 生成 (new) しないと使えない

```
Person tarou = new Person();
```

よくある間違い

- フィールドにアクセスしていない

```
Person tarou = new Person();  
tarou = "理工太郎";  
tarou.age = 18;
```

Personクラス型の変数に、
文字列を代入しようとしている

Person型に代入できるのは Person 型だけ

よくある間違い

- 出力時にフィールドを表示していない

```
System.out.println(tarou + "(" + tarou.age + "歳)");
```

謎の文字列が表示される

|Person@7a81197d(18歳)

クラス名@~~~~ の文字列が表示されたときは、
．で繋ぐものが不足していないかをよく確認すること

実際は謎でも何でもなく、オブジェクト指向を学んでいくと理解出来るようになる
(少なくとも、基礎の範疇で理解するのは難しい)

よくある間違い

- クラスの中にクラスを宣言している

```
public class Main07p1 {  
    public static void main(String[] args) {  
        class Person {  
            String name;  
            int age;  
        }  
        Person tarou = new Person(); //Person型のtarou  
        tarou.name = "理工太郎";  
        tarou.age = 18;  
        System.out.println(tarou.name + "(" + tarou.age + "歳)");  
    }  
}
```

メソッド内にクラスを作らない！

エラーにはならないが、プログラミング基礎 I では禁止
(こうすることによるメリット・デメリットを理解していて、
自分の言葉で説明出来るなら許可する)

クラス型変数の便利な使い方

- 2人目の学生情報を設定

- まず、 Person型の変数を宣言（学生には人が必要）

```
Person hanako = new Person();  
hanako.name = "九産花子";  
hanako.age = 19;
```

- Student型の変数を宣言し、フィールドを設定

```
Student student2 = new Student();  
student2.studentNumber = "24RS999";  
student2.person = hanako;  
student2.subject1 = math;  
student2.subject2 = english;
```

- 科目は同じなので再利用する

Subject型の値を変更（教室変更）

- mathのclassNameを "0A教室1" に変更

```
math.className = "0A教室1";
```

- student1とstudent2 の情報を再出力

```
System.out.print(student1.studentNumber + " " + student1.person.name + "の");  
System.out.println(student1.subject1.name + "の教室:" + student1.subject1.className);  
  
System.out.print(student2.studentNumber + " " + student2.person.name + "の");  
System.out.println(student2.subject1.name + "の教室:" + student2.subject1.className);
```

```
25RS999 理工太郎の数学の教室:12107  
24RS999 九産花子の数学の教室:12107  
25RS999 理工太郎の数学の教室:0A教室1  
24RS999 九産花子の数学の教室:0A教室1
```

mathの教室を変更しただけなのに、
学生の履修情報の方も変更された

student1(Student型のインスタンス)

studentNumber	25RS999
person	
subject1	
subject2	

math(Subject型のインスタンス)

name	数学
credits	2
className	12107

同じデータを参照している

student2(Student型のインスタンス)

studentNumber	24RS999
person	
subject1	
subject2	

english(Subject型のインスタンス)

name	英語
credits	1
className	S202

student1(Student型のインスタンス)

studentNumber	25RS999
person	
subject1	
subject2	

math(Subject型のインスタンス)

name	数学
credits	2
className	OA教室1

student2(Student型のインスタンス)

studentNumber	24RS999
person	
subject1	
subject2	

同じデータを参照している
から、書き換えると
student1,2の数学の教室情報も変わる

name	英語
credits	1
className	S202

まとめ（クラス）

- クラス

- ものの性質や機能をまとめた設計書

- 例：人には名前と年齢がある
 - “人”クラス内に性質をまとめたフィールドを作る

- 作成したクラスをもとにオブジェクトを作る

- 例：クラスを使って、人を作る
 - 作った人には名前と年齢をつける



人型のオブジェクト p1
name:Taro
age:20



人型のオブジェクト p2
name:Hanako
age:23



人型のオブジェクト p3
name:Ichiro
age:25