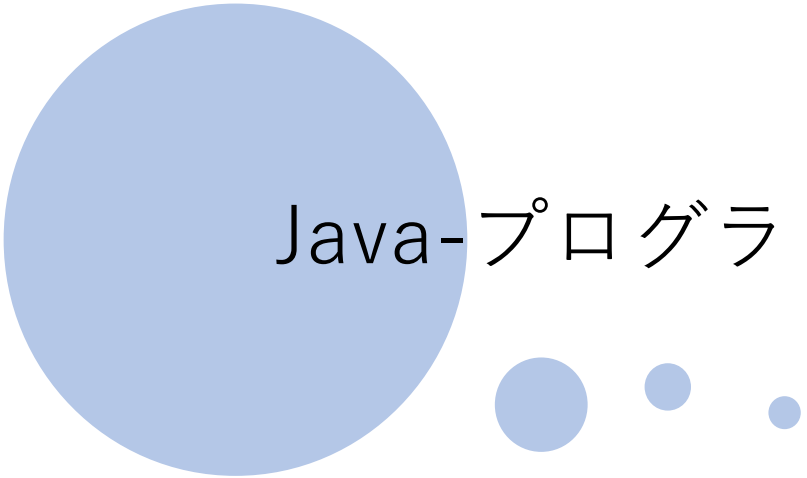
2016/4/18 スタートアップゼミ#1



# Java-プログラミングの基礎

M1 庄司惟

# 0. 目次

**-基本編-**

1. JavaとEclipseの使いどころと特徴
2. 基礎用語・基本概念
3. 作法
4. 基礎計算（基礎中の基礎）
5. PPデータ集計用のプロジェクト構成を確認（実践）

**-実用編-**

1. 九九練習プログラムで基礎確認
2. 時間計測方法
3. データの格納手法
4. 値の探索と並び替え（ソート）
5. PPデータから徒歩データのみ抽出（実践）
6. 計算量

# 1. JavaとEclipseの使いどころと特徴

## 使用例

・Excelでは開けない**巨大**ファイル処理・加工
例）１億行データのソート ←1048576行までしかむり

・単純作業の**繰り返し**
例）複数ファイルのラベルの英語化

・Excelでは扱えない**複雑**処理
例）マップマッチング，データ変換

# 1. JavaとEclipseの使いどころと特徴

**Javaって何？他のと違って？**

・プログラミング言語（あたりまえ）
・オブジェクト指向言語
・型の制約が厳しい
・コンピュータ機種の違いによる移植が不要

**Eclipseってなに？**

・統合開発環境（Integrated Development Environment :IDE）
・ソースコードの編集，プログラム実行，デバッグ
　を一つの環境で行う．

# 2. 基礎用語・基礎概念

**プロジェクト**

・プロジェクトとは？
⇒一つのプログラムの基本単位
⇒例）MapMatching.java, DrmChange.java

・プロジェクトの作成
⇒ファイル→新規→Javaプロジェクト

・実行
⇒Ctrl+F11で実行

# 2. 基礎用語・基礎概念

## クラス

・クラスとは？
⇒関連情報を一つにまとめたもの

・クラスの作成
⇒ファイル→新規→クラス
※public static void main(String[] args)に☑

**クラスの定義**

```
Public class Kamoku {
        String name; //科目名
        int score; //点数
}
```

**Kamokuクラスのインスタンスの作成**

```
Kamoku kokugo = new Kamoku (); //科目を一つ作る
kokugo.name = “国語”; //科目名を国語にする
kokugo.score = 63; //点数を63点にする
```

kokugoという一つの変数さえ持っていれば,
いつも「国語」という名前と「63点」という点数を引き出せる！

# 2. 基礎用語・基礎概念

## クラスとインスタンス

例

**“学生”クラス**

**“学生”インスタンス**
名前：“Yばし”
学年：1
学籍番号：0815
出身：“関東”
性別：“男“

**“学生”インスタンス**
名前：“Tきー”
学年：2
学籍番号：1116
出身：“九州”
性別：“男“

**“学生”インスタンス**
名前：“Eぽよ”
学年：2
学籍番号：1123
出身：“近畿”
性別：“女“

研究室のプログラムの例だと、

**Nodeクラス・・・プログラム中でノードデータを格納する**

**Nodeインスタンス**
ノードID：0
緯度：35.75
経度：36.64
最短経路探索済：0

**Nodeインスタンス**
ノードID：1
緯度：35.99
経度：36.32
最短経路探索済：0

**Nodeインスタンス**
ノードID：2
緯度：35.88
経度：36.09
最短経路探索済：0

# 2. 基礎用語・基礎概念

**メソッド**

・よく使うデータの挿入や抽出，計算はメソッドにする

```
public class Main {
  public static void main(String args[]){
    double a = 1.5;
    double b = 1.5;
    double ans = product(a,b);    //aとbをproductメソッドを使って計算
    System.out.println(ans);      //答えを表示
  }

          //ここからがメソッド、かけ算をするメソッドを作る
  private static double product(double x, double y){
          //返り値の型がdouble、引数にdoubleの変数xとyをとるという意味
    return (x * y);    //returnで返り値を表す
  }
}
```

Mainクラス

メソッド

# 3. 作法

・変数，インスタンス，関数名のつけ方
⇒最初は必ず**小文字**で始める.
⇒クラス名は**大文字**で始める
⇒単語の区切りは大文字or ”_” でつなぐ.
例) getData()，MAX_BUFFER_SPACE

・演算子の前後はスペースを空ける.
例）y=a*x+b;　→　y = a * x + b;

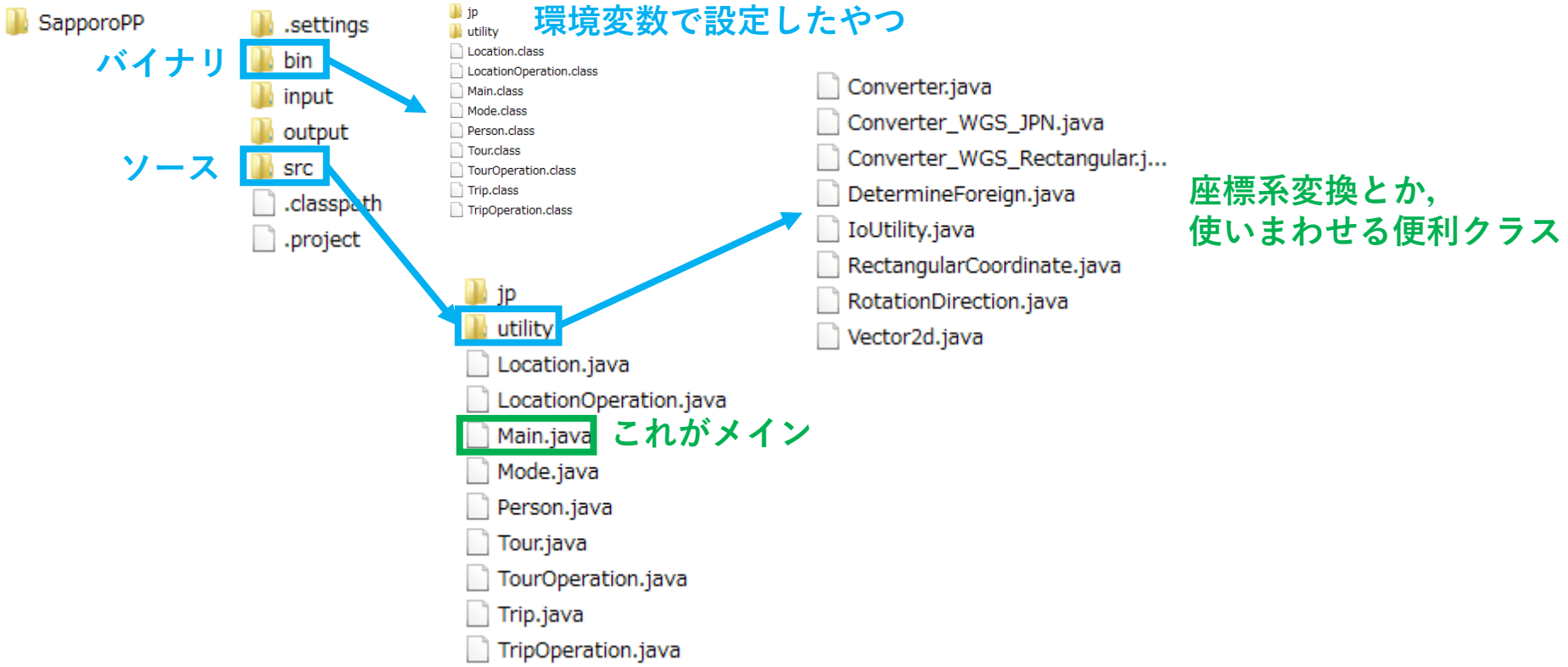
# 4. 基礎中の基礎（再確認）

1. 変数の型（String, int, double,…）
2. セミコロン（;）で指示を終了
3. 基礎演算子（+,-,*,/,%,…）
4. 結果の表示（System.out.println()）
5. 制御構文（if ~ else, for, while）
6. 比較演算子（==,!=,>,>=,<=,…）

※　=は代入を表す

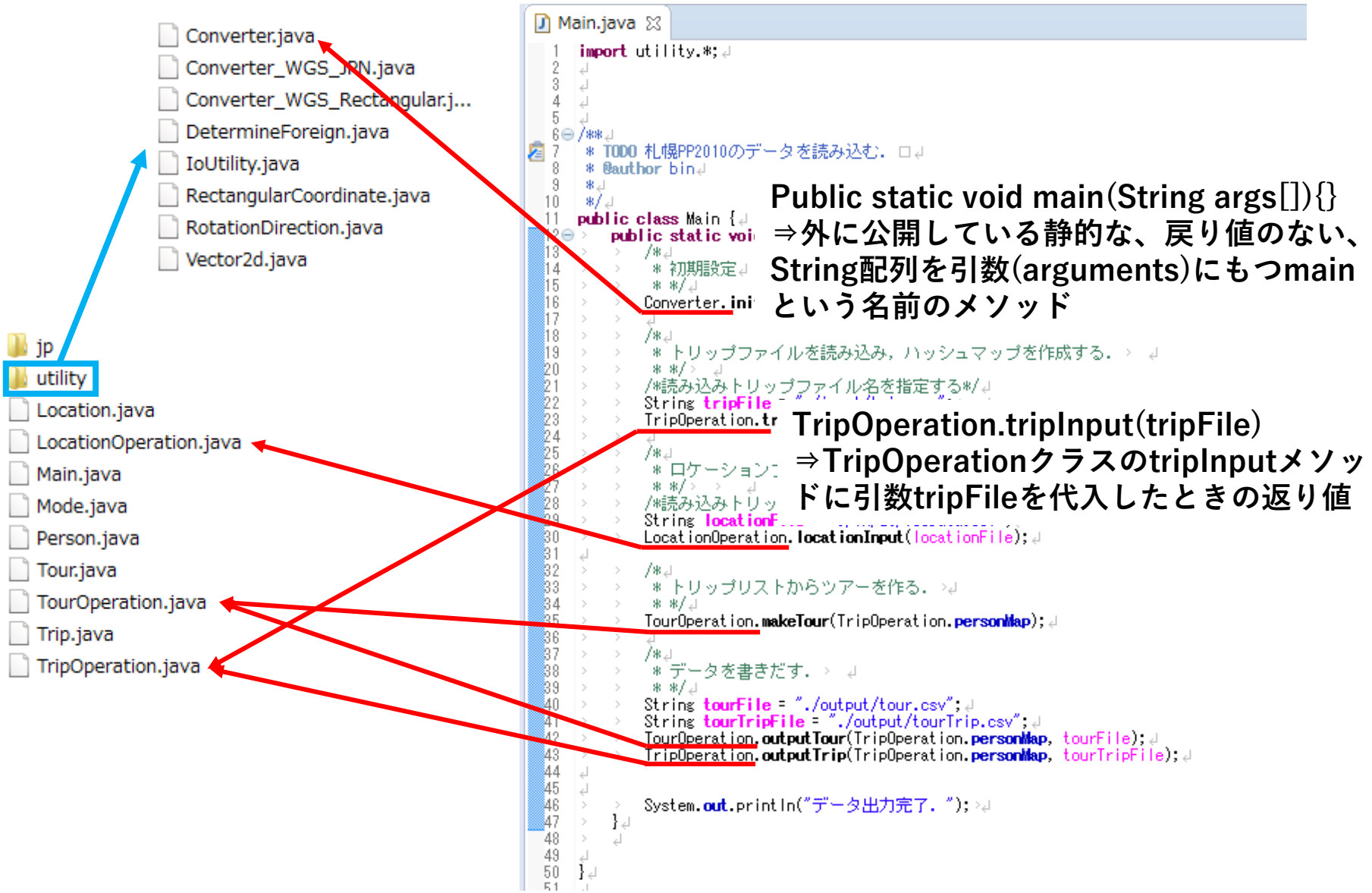
# 3 Wikiに上がってるjavaプロジェクトの理解

## 例）PPデータ集計プログラム(SapporoPP)



# 3 Wikiに上がってるjavaプロジェクトの理解

## 例）PPデータ集計プログラム(SapporoPP)



# 実用編に入ります

これからの流れ

1) 九九練習プログラムで基礎確認
2) 時間計測方法
3) データの格納手法
4) 値の探索と並び替え（ソート）
5) PPデータから徒歩データのみ抽出（実践）
6) 計算量

# 3 九九練習プログラムで基礎確認

・九九の問題を出題し，userが解答すると正誤を表示
・順番に10問繰り返し，最後にまとめた正答率を表示

これから九九の問題を10問出します 表示→System.out.println()
[第一問]　4 × 7 = ?　1～9の乱整数→(int)(Math.random()*9)+1
**User > 28**　入力→BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
正解です　条件→if文
[第二問]　9 × 8 = ?
**User > 74**
不正解です
…
[第十問]　4 × 7 = ?
**User > 28**
正解です

繰り返し→for文
正答数をカウントしておく

10問中7問正解で，正答率は70%です.
お疲れさまでした.

Java言語プログラミングレッスン（上）P74より

# 2. 計算時間計測

## currentTimeMills()

・1970年1月1日からの経過時間をミリ秒単位で取得
・計算開始時と終了時の時間差で，計算時間計測
・long型を使用する

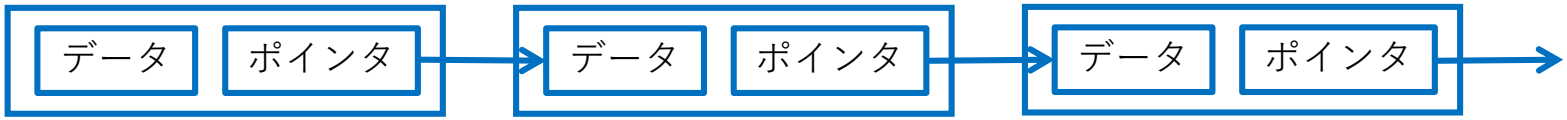
```
public class Main {
   public static void main(String[] args) {
      long start = System.currentTimeMillis();          //計算開始時の時刻を取得

      for (int i = 0; i < 1000000; i++){     //計算の内容
         System.out.println(i + 1);          //ここでは1から100万までの数字を表示させる
      }

     long finish = System.currentTimeMillis();          //計算終了時の時刻を取得
     double duration = (finish - start) / 1000;         //ミリ秒から秒に変換
     System.out.println("計算時間:" + duration + "秒");   //計算時間を表示
   }
}
```

```
1
:
999999
1000000
計算時間:16.0秒
```

# 3. データの格納

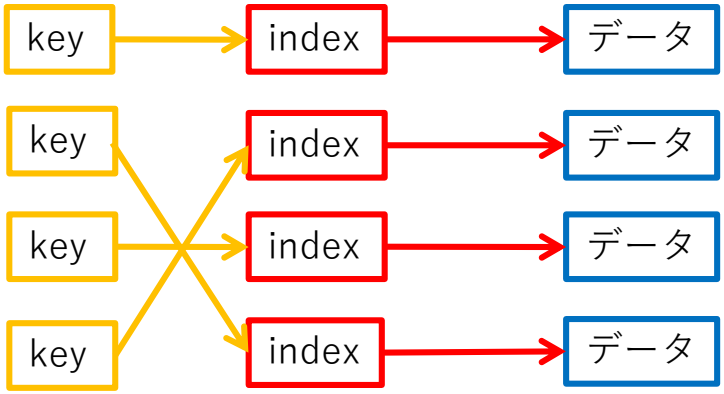
**ArrayList**…線形リストを実装したクラス

線形リスト…データと，次のデータを指すポインタの配列




**HashMap**…ハッシュテーブルを実装したクラス

ハッシュテーブル…キーをもとにしたハッシュ値が指すデータから
キーとデータの対応付けをしたデータ構造



# 3. データの格納

つまるところ…

**Arraylist**：順番と要素で管理

| | |
|---|---|
| 0 | 日本 |
| 1 | カナダ |
| 2 | アメリカ |
| 3 | ドイツ |
| 4 | 中国 |
| : | : |

**Hashmap**：キーと要素で管理

| | |
|---|---|
| Japan | 日本 |
| Canada | カナダ |
| USA | アメリカ |
| Germany | ドイツ |
| China | 中国 |
| : | : |

**使い方**

①宣言

- List<ｸﾗｽ>ｲﾝｽﾀﾝｽ名 ＝ new ArrayList<ｸﾗｽ>();
- Map<ｷｰ,ｸﾗｽ>ｲﾝｽﾀﾝｽ名 ＝ new HashMap<ｷｰ,ｸﾗｽ>();

②データの追加

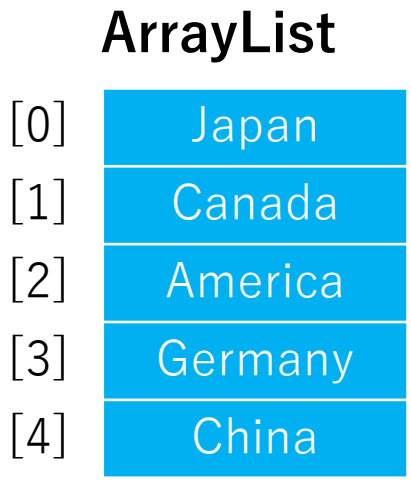
- ｲﾝｽﾀﾝｽ名.add(要素);
- ｲﾝｽﾀﾝｽ名.put(ｷｰ,要素);

③データの取り出し

- ｲﾝｽﾀﾝｽ名.add(順番);
- ｲﾝｽﾀﾝｽ名.get(ｷｰ);

# 4. 値の探索と並べ替え

探索：”Germany”は何番目にあるか？
ソート：英語辞書の順番に並べ替えよ.

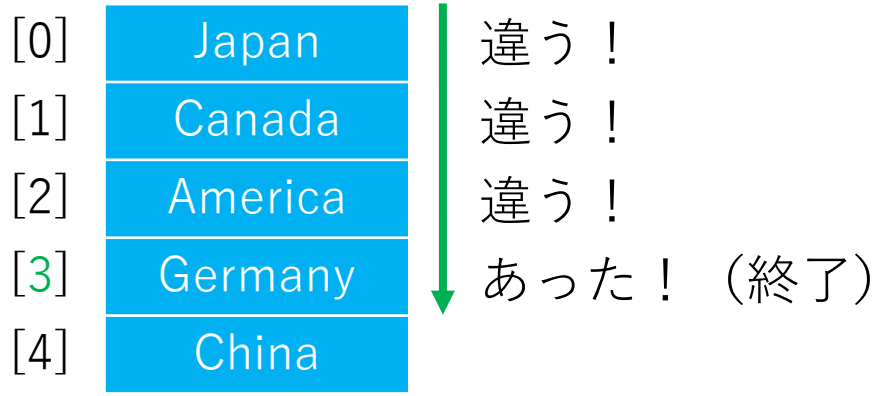


# 4. 値の探索と並べ替え

## 線形探索

・順番に配列の中身を調べていって，答えが見つかれば終了

例）”Germany”は何番目にあるか？



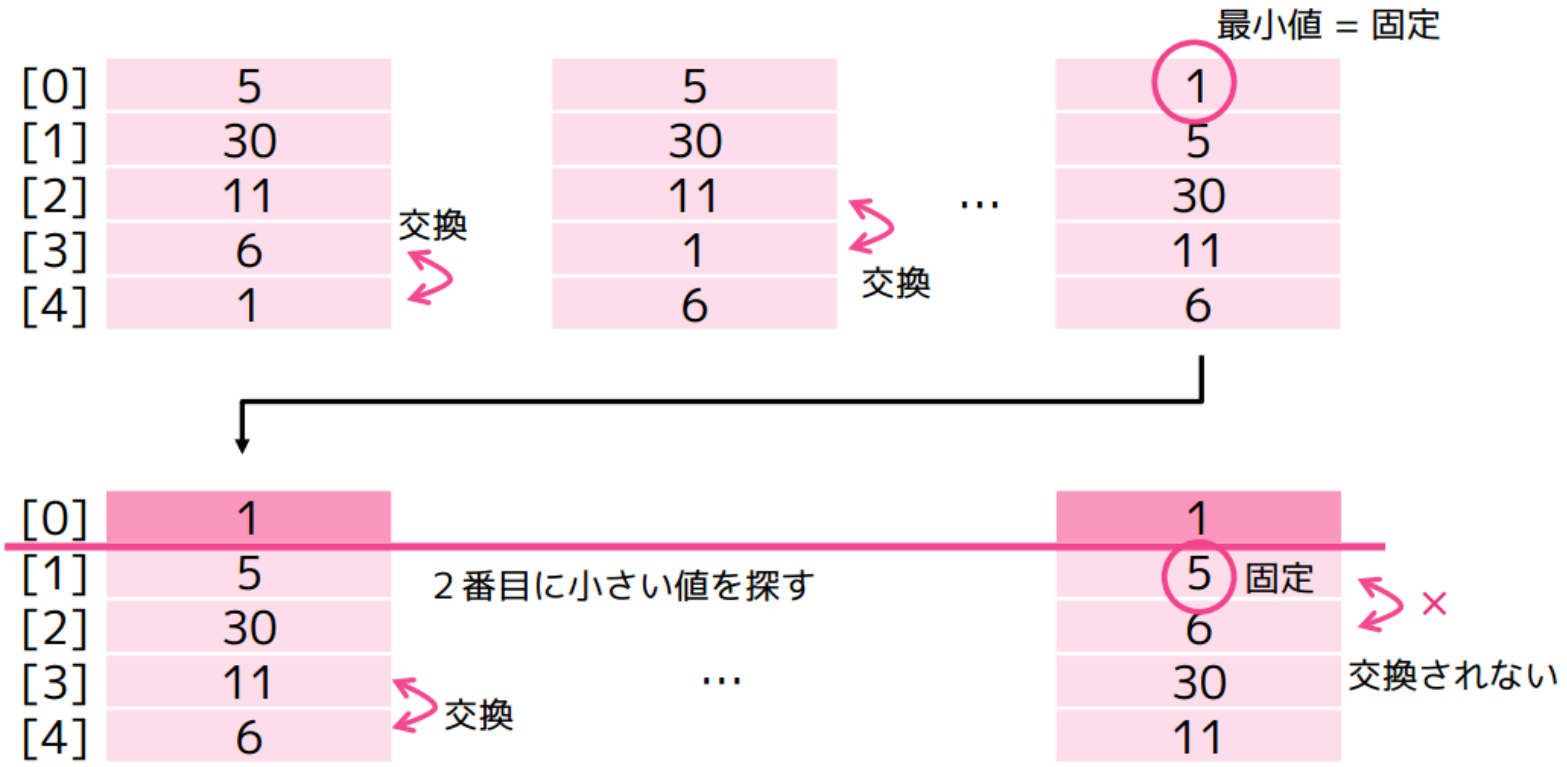

（答）3 + 1 = 4（番目）

# 4. 値の探索と並べ替え

## バブルソート

・上の要素と比較し，上の方が大きければ交換
・最小値から順番に決定して（浮かんで）いく

例）数値を小さい順に並べよ



# 5. ファイル読み込みと書き込み

**BufferedReader**

・ファイルの入力
・readLine()で1行ごとに読み込む

**PrintWriter**

・ファイルの出力
・printlnで1行ごとに書き込む

型は大丈夫ですよね？
・String
・float(32), double(64)
・integer(32), long(64)

※どちらも,
・tryとcatchで例外処理を行わねばならない
・close()で必ずファイルを閉じる
・String型で入出力が行われる
→型の変換が必要（次ページ）

# 型変換

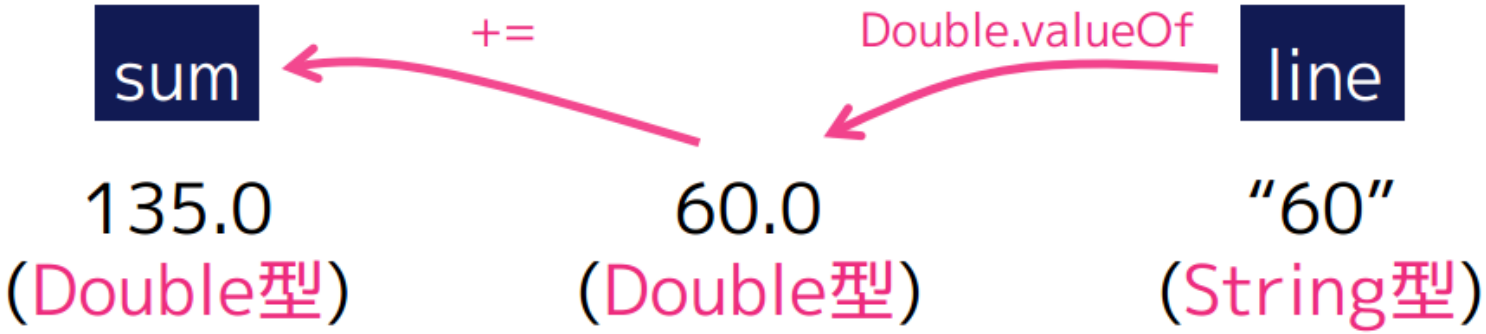
**valueOf**　※Oはゼロじゃなくオー

数値⇔文字列　の変換

例）String.valueOf(数値)　　⇒　文字列

例）Double.valueOf(文字列)　⇒　数値(Double型)

```
11行目　sum += Double.valueOf(line);
```



# 5. ファイル読み込みと書き込み

## 実践（読み取り→合計→書き込み）

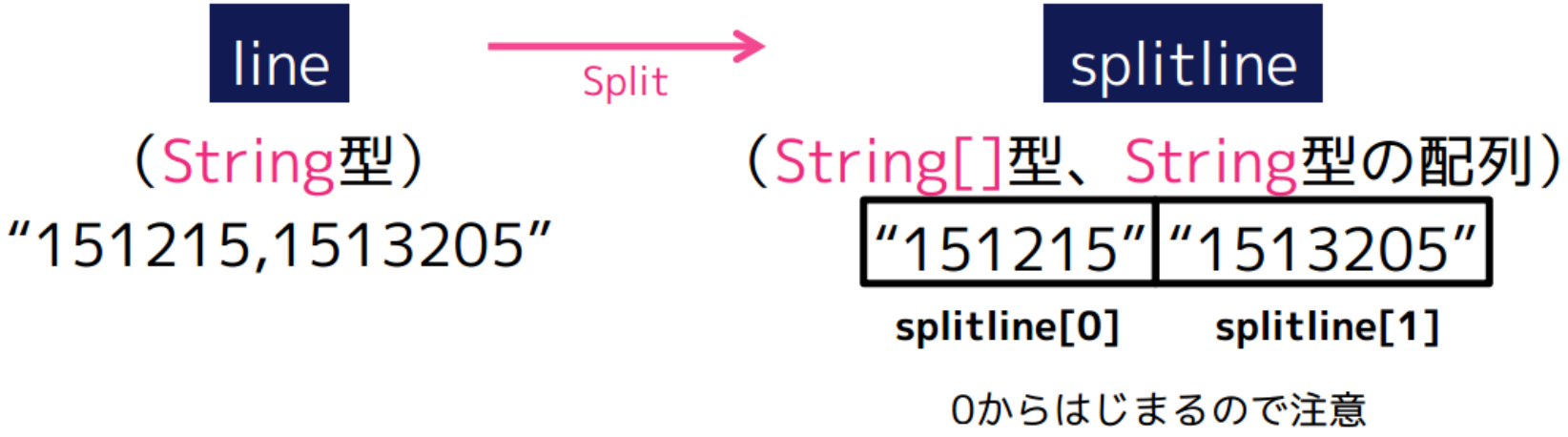
```
import java.io.*;                                    //java.ioパッケージを使う

public class Main {
  public static void main(String[] args) {
例外  try {                                          //データをうまく入出力できるとき
        String inputfile = "./input/input1.csv";          //インプットファイル名
  読込  BufferedReader br = new BufferedReader(new FileReader(inputfile));
        String line = null;                               //1行ごとに読み込む変数を用意
        double sum = 0;                                   //合計値を足していく変数を用意
        while ((line = br.readLine()) != null) {          //最終行になるまで読み込む
          sum += Double.valueOf(line);               //各行の内容をint形式にしてsumに足す
        }
    閉  br.close();

        String outputfile = "./output/output1.txt";       //アウトプットファイル名
  書込  PrintWriter pw = new PrintWriter(new FileWriter(outputfile));
        pw.println("合計:" + String.valueOf(sum));        //sumをString形式にして書き込み
    閉  pw.close();
      }
例外  catch( IOException e ) {                       //データを入出力ができなかったとき
        System.out.println("データ入出力失敗");
      }
    }
}
```

# 5. ファイル読み込みと書き込み

複数列csvファイルの読み込み
⇒コンマ区切りで読み取る

**split()**…特定の文字で区切って配列化

```
line=br.readLine()//1行分読み込み
String[] splitline = line.split(",");
```



# 6. 計算量

**金庫を破ろう**

・n桁の暗証番号を線形探索で解く場合

・n=4であれば，10000通り


**コンピュータの計算能力と計算量の比較**

Core i7-4770Kプロセッサ(3.8cm×3.8cm）は一秒間に164,180,000,000回の単純計算ができる.
これを地球上に敷き詰めると（3.54*10^9個)地球全体の計算能力は5.81*10^28回/秒. で以下概算すると

| **n** | 1 | 2 | 5 | 10 | 20 | 50 | 100 |
|---|---|---|---|---|---|---|---|
| **最悪の手数** | 10 | 100 | 10^5 | 10^10 | 10^20 | 10^50 | 10^100 |
| **計算時間** | 1.6*19^-28 秒 | 1.6*10^-27 秒 | 1.6*10^-24 秒 | 1.6*10^-19 秒 | 1.6*10^-9 秒 | 3520 宙齢 | 3.52*10^53 宙齢 |

**宙齢**は、宇宙の塵から地球ができてこのかた現在までの年数 (150 億年) を表す単位

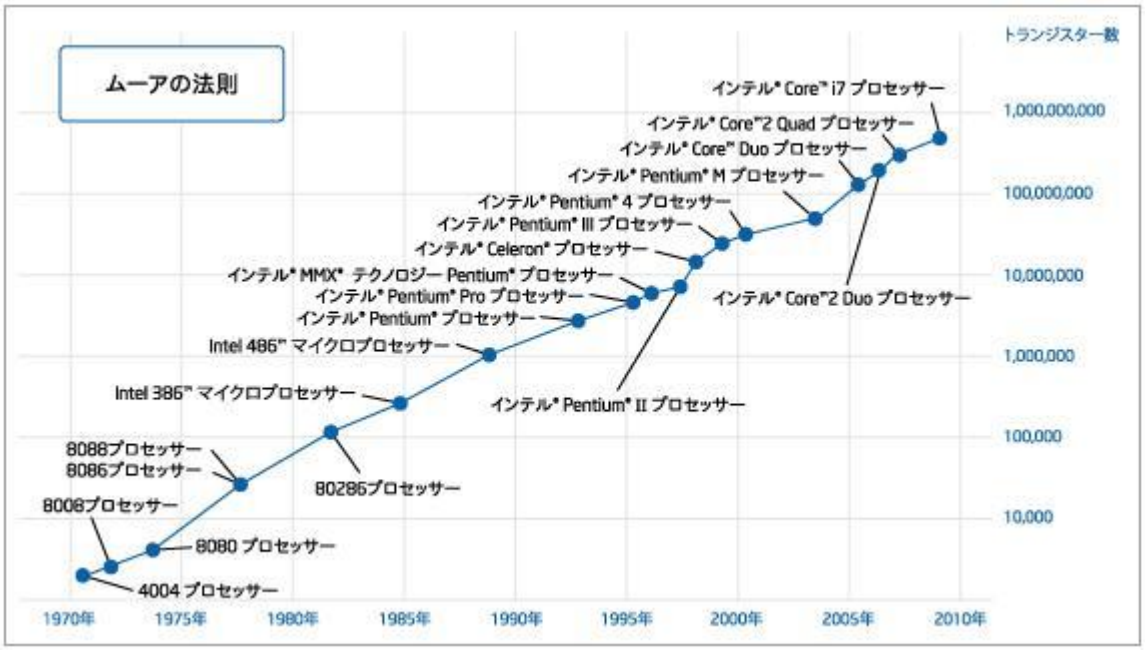
今後，Corei7の速度が10000倍になったくらいでは到底解決できない！

http://www.coins.tsukuba.ac.jp/~tomohiro/literacy/2015/lecture/complexity.html

# 6. 計算量

**ムーアの法則の終焉**が騒がれていますが…

ムーアの法則
半導体集積回路のトランジスタ数は2年ごとに2倍になる




計算速度向上にも限界が見えてきている
→これらの問題を解くのに，アルゴリズムが重要になってくる

http://www.coins.tsukuba.ac.jp/~tomohiro/literacy/2015/lecture/complexity.html

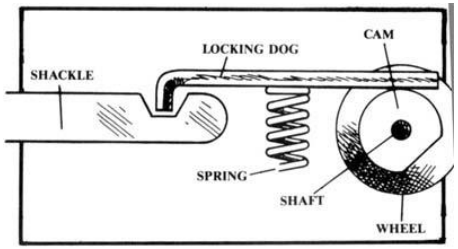
# 6. 計算量

**金庫を破ろう**

・カギ屋さんはセサミロックデコーダ（特殊形状の金属板）を使って一瞬で開けてしまう.

・番号の当たり外れが一桁ずつわかる！

・人間は一秒間に一つの番号を試せるとして,




| n | 1 | 2 | 5 | 10 | 20 | 50 | 100 |
|---|---|---|---|---|---|---|---|
| **デコーダ無** | 10秒 | 100秒 | 27.8時間 | 316.9年 | 3.16兆年 | 2.11*10^32 宙齢 | 2.11*10^82 宙齢 |
| **デコーダ有** | 10秒 | 20秒 | 50秒 | 1分40秒 | 3分20秒 | 8分20秒 | 16分40秒 |

nが大きくなるほどその差は顕著！

**計算量**は　デコーダ無：10^n⇒O(10^n)（指数時間）　＝爆発

デコータ有：10*n⇒O(n)（多項式時間）＝解ける

ビッグ・オー記法

http://www.coins.tsukuba.ac.jp/~tomohiro/literacy/2015/lecture/complexity.html

# 計算速度短縮のポイント

・適切な**データ構造**を用いてデータをメモリにいれる.
　これにより**繰り返し**を省く
　　例）配列，ハッシュテーブルを用いてfor文減らす.
　　二分木やヒープを用いてforループ軽減.

・もう使わないデータを無駄に残しておかない（**メモリ節約**）
　　⇒配列，マップのクリア

# 課題(inputデータはwikiから各自DL)

表示…コンソールに表示
出力…ファイルに出力

1. ”Hello!World!”と表示する
2. 1~100までの数字を表示する
3. 1~100までの数字を一行で表示する
4. 1~100のうち素数のみを順番に表示する
5. 以下の式を満たす最小のx,yをそれぞれ求める.

$$\left(\sum_{i=1}^{x} i\right) > 100, (y!) > 1000000$$

6. input1.csvのデータの合計・平均・標準偏差を計算し output1.txtとして出力，さらに計算時間を表示する.

**input1.csv**

```
75
60
80
45.5
37
55
90
16.2
75
19
```

**output1.txt**

```
合計:552.7
平均:55.27
標準偏差:25.58203
```

# 課題(inputデータはwikiから各自DL)

7. input2.csvの一列目と二列目の和と差を各行で計算し，output2.csvとして一行目に和，二行目に差を出力する．

**input2.csv**

```
151215,1513205
4564258,151
672842,5446
3542415,6545
84542,1215
```

**output2.csv**

```
1664420,1361990
4564409,4564258
678288,667396
3548960,3535870
85757,83327
```

ヒント：Math.abs

8. input2-2.csvの一行目と二行目について問題7. と同様に計算・出力する．ただし三行目がNGの場合は出力しないこと．

**input2-2.csv**

```
151215,1513205,OK
4564258,151,OK
672842,5446,OK
3542415,6545,NG
84542,1215,OK
```

**output2-2.csv**

```
1664420,1361990
4564409,4564258
678288,667396
85757,83327
```

ヒント：文字列比較は“equals()”

# 課題(inputデータはwikiから各自DL)

9. input3.csvからOD表(output3.csv)を作る
   ※各O-D組ごとに拡大係数を合計
   ※行が同一O，列が同一D
   ※ （発展）代表手段別のOD表も作る

**input3.csv**

```
出発,到着,手段,目的,拡大係数
0,0,鉄道,業務,0083
0,0,鉄道,業務,0083
0,2,鉄道,業務,0083
2,0,鉄道,帰宅,0083
0,0,徒歩,買い物,0037
0,0,徒歩,帰宅,0037
0,0,鉄道,通勤,0047
:
4,4,自転車,帰宅,0092
```

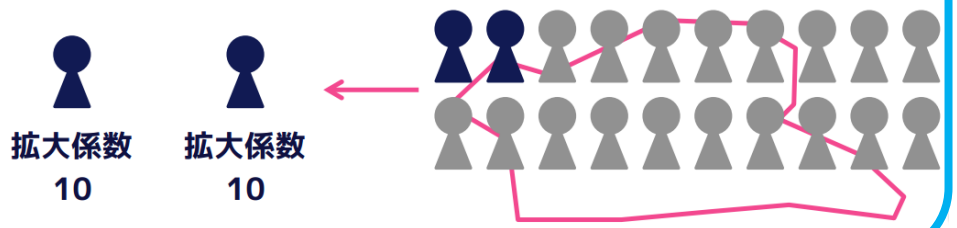
**output3.csv**

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | ? | ? | ? | ? | ? | ? |
| 1 | ? | ? | ? | ? | ? | ? |
| 2 | ? | ? | ? | ? | ? | ? |
| 3 | ? | ? | ? | ? | ? | ? |
| 4 | ? | ? | ? | ? | ? | ? |
| 5 | ? | ? | ? | ? | ? | ? |

**拡大係数**
⇒何人分のトリップを代表しているか

**男性、20～25歳、地域Aの取得サンプル・・・2人**

**男性、20～25歳、地域Aの居住者・・・20人**



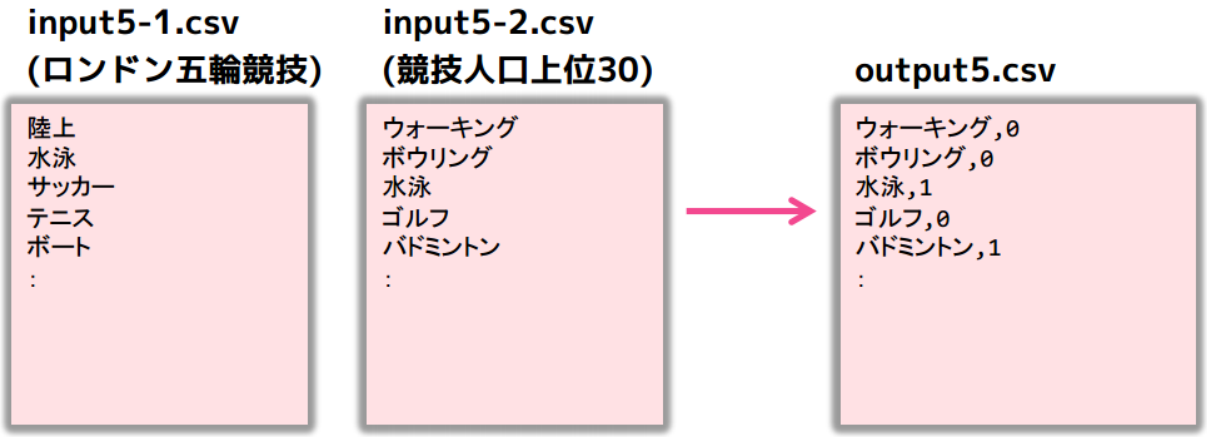

ヒント：int[][] od = new int[6][6] でOD行列を初期化

# 課題(inputデータはwikiから各自DL)

10. 二地点の緯度経度から距離を計算するメソッドを作成
    ※引数は４つ（Aの緯度，Aの経度，Bの緯度，Bの経度）
    ※緯度経度と距離の関係はぐぐる
    ※地球を完全な球とみなしてもよいかも？
    ※東京駅(35.681143,139.767208)から
    横浜駅(35.466193,139.622498)の距離
    ＝27280mで答え合わせして確認する

ヒント：Math.toRadians, Math.sqrtなど

11. 競技人口上位30のスポーツがロンドン五輪競技に含まれるかどうかを調べる．（入っていたら1,そうでなければ0）




ヒント：ArrayList, equals()

# 課題(inputデータはwikiから各自DL)

12. input6-2.csvの位置データ一つ一つに対して，最寄り駅(input6-1.csv)を求めて出力する．

**input6-1.csv**
**(山手線内駅データ)**

```
東京,35.6813,139.7661
上野,35.7137,139.7770
有楽町,35.6754,139.7638
新橋,35.6661,139.7585
浜松町,35.6553,139.7571
```

**input6-2.csv**
**(位置データ)**

```
35.6921,139.7515
35.6435,139.7204
35.6242,139.7495
35.7122,139.7354
35.6513,139.7264
:
```

→

**output6.csv**

```
35.6921,139.7515, 飯田橋
35.6435,139.7204, 恵比寿
35.6242,139.7495, 品川
35.7122,139.7354, 飯田橋
35.6513,139.7264, 恵比寿
:
```

ヒント：クラスを自分で定義する（Stationクラス）
ヒント：課題10のメソッドを使える

# 課題(inputデータはwikiから各自DL)

13. 配列 : int[] a = {14,59,1,20,37,90,22}　について,
　　(1)バブルソートアルゴリズムを使って並び替え
　　(2)線形探索で「22」が何番目に来たかを求めよ