プログラミングゼミ

(第5回スタートアップゼミ？)

2012年6月26日(火)

M1 伊藤 創太

Java編

計算時間の計測

currentTimeMillis()メソッド

・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・終了時の取得時間の差で、計算時間を計測
・プログラム全体や、部分ごとの計算時間計測も可能
・1000ミリ秒 = 1秒
・long型を使用

long start = System.currentTimeMillis();
start･･･1340418208076（2012年6月23日11:30の場合）

計算時間の計測

currentTimeMillis()メソッド

コード例：

```
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
   }
}
```

実行結果：

```
1
:
999999
1000000
計算時間:16.0秒
```

合計・平均・標準偏差

データの集計 – 合計を求める

・データを集計したり、計算したりしたいことがあるよね
・大きいデータだと、Excelで開けないこともあるよね
・簡単な計算はサクッとできるようになりたい

インプットデータ：

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

こうしたい

アウトプットデータ：

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

合計・平均・標準偏差

データの集計 – 合計を求める

・ここでは、合計の求め方だけ、例を示す

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

public class Main {
  public static void main(String[] args) {
    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
      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
      }
      br.close();

      String outputfile = “./output/output1.txt”;       //アウトプットファイル名
      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
      pw.close();
    }
    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");
    }
  }
}
```

合計・平均・標準偏差

valueOf

・数値→文字列、文字列→数値の変換

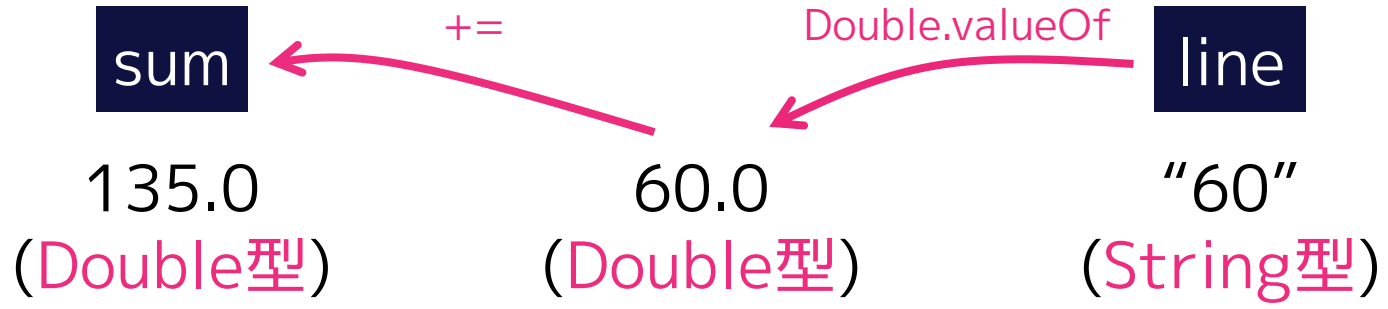
・例1：String.valueOf(数値)

　・・・数値を文字列(String)に変換

・例2：Double.valueOf(文字列)

　・・・文字列(String)を数値(Double)に変換

11行目　`sum += Double.valueOf(line);`



合計・平均・標準偏差

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

・String型で入出力が行われる

・tryとcatchで例外処理を書かなければならない

・close()で閉じるのを忘れずに

詳細はスタートアップゼミ第3回参照のこと。

課題1

データの集計 – 合計・平均・標準偏差

・合計と平均と標準偏差を求めるプログラムを作れ

・どんな変数を用意するべきか

・標準偏差 $$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{x} - x_i)^2}$$

・ヒント：Math.sqrt

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1’

データの集計 – 合計・平均・標準偏差

・課題1の計算に要した時間を計測して、コンソールに表示させよ。

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
```

和と差

データの集計 – 足し合わせ

・データの和や差をとりたいことがあるよね
・大きいデータだと、Excelで開けないこともあるよね
・簡単な計算はサクッとできるようになりたい

インプットデータ：

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

こうしたい

アウトプットデータ：

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

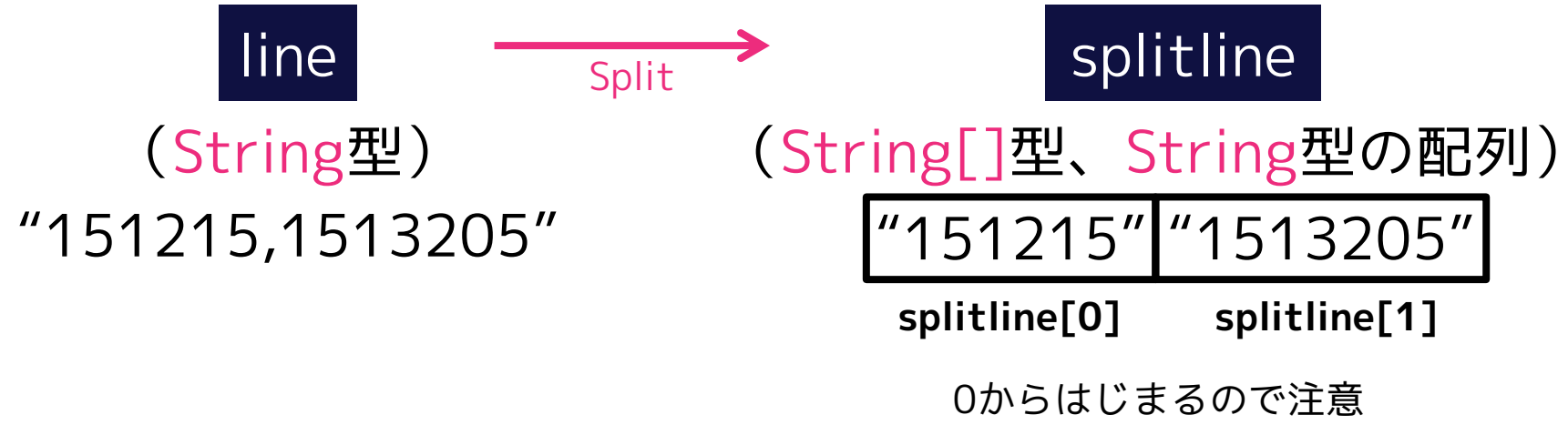
和と差

複数列のcsvファイルの読み込み – split

・特定の文字で区切って配列にしてくれる

例

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



課題2

データの集計 – 複数データの和と差

・1列目と2列目の和と差を各行について求めよ
・アウトプットデータの1行目に和、2行目に差を出力
・各行の和と差の値を格納する方法が必要
・ヒント：Math.abs、println(何か + “,” + 何か)

input2.csv

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

output2.csv

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

集計・カウント

データの集計 – カウント

・属性ごとに集計したいことがあるよね
・大きいデータだと、Excelで開けないこともあるよね
・簡単な計算はサクッとできるようになりたい

インプットデータ：

```
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
```

出発ゾーンごとに
発生交通量を
集計したい

アウトプットデータ：

```
0,34887185
1,19043082
2,12088410
3,12743172
4,3425779
5,2642548
```

※データは実際の東京都市圏PTデータを加工したもの
ゾーンは、0:東京、1:神奈川、2:埼玉、3:千葉、4:茨城、5:東京都市圏外

(補足)PTデータの拡大係数について

層別拡大

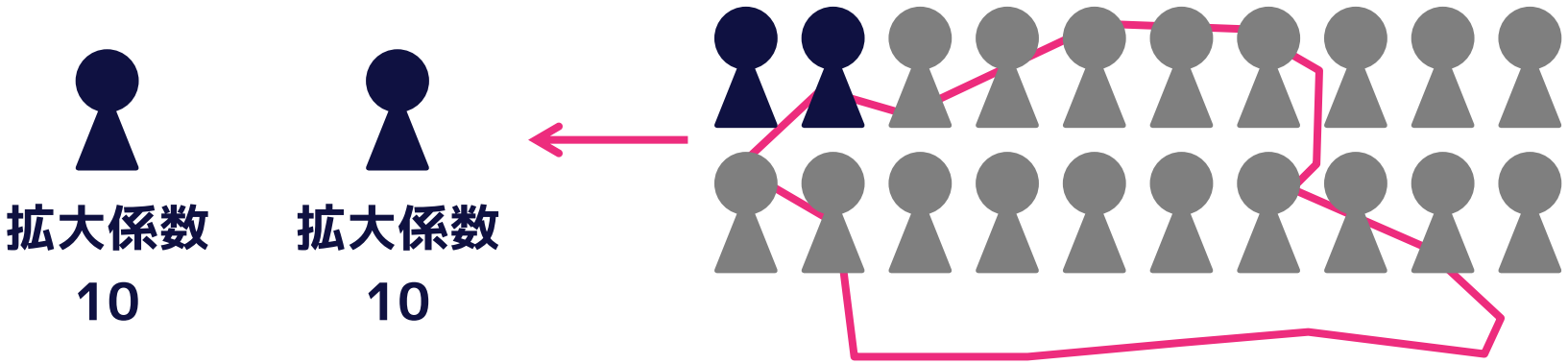
・居住地・年齢階層・性別で層別に拡大

・抽出データから実際のトリップ数に拡大する

・何人分のトリップを代表しているかという意味

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

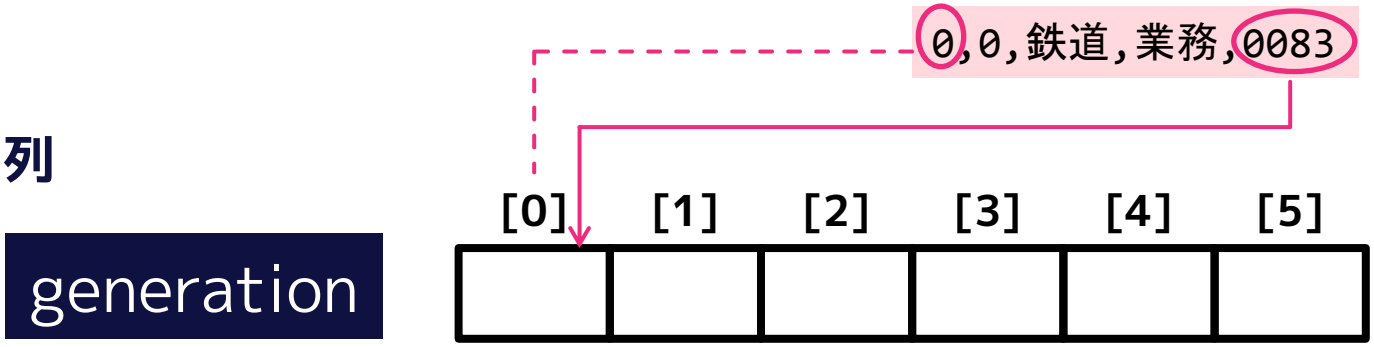


集計・カウント

データの集計 – 発生交通量の集計（前半）

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

public class Main {
  public static void main(String[] args) {
    try {                                           //データをうまく入出力できるとき
      String inputfile = "./input/input3.csv";      //インプットファイル名
      BufferedReader br = new BufferedReader(new FileReader(inputfile));
      String line = null;                           //1行ごとに読み込む変数を用意
      int[] generation = new int[6];                //要素6つの配列用意、各ゾーンの発生量が入る
      while ((line = br.readLine()) != null) {      //最終行になるまで読み込む
        String[] splitline = line.split(",");       //カンマ区切りで分割
        generation[Integer.valueOf(splitline[0])] += Integer.valueOf(splitline[4]);
                                  //ゾーン(1列目)に対して拡大係数(5列目)をint型にして足す
      }
      br.close();
```



データの集計 – 発生交通量の集計（後半）

```
      String outputfile = “./output/output3.txt”;      //アウトプットファイル名
      PrintWriter pw = new PrintWriter(new FileWriter(outputfile));
      for (int i = 0; i < 6; i++) {          //ゾーン0～5の集計結果を出力したい
        pw.println(i + “:” + String.valueOf(generation[i]));
                                  //ゾーンiの発生交通量を書き出し
      }
      pw.close();
    }
    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");
    }
  }
}
```

課題3

データの集計 – OD表を作ろう

・出発・到着ゾーンごとに拡大係数を合計せよ
・表の行方向を出発、列方向を到着とする
・余力があれば代表交通手段別のODも求めよ
・ヒント：int[][] od = new int[6][6]でOD行列を初期化

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	?	?	?	?	?	?

メソッド

メソッド

・以前Rで扱った関数のようなもの

・戻り値・引数を指定する

・よく使うデータの挿入や抽出、計算などはメソッドに。

```
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
  }
}
```

課題4

メソッド – 2地点の緯度経度から距離を計算

・4つの引数（Aの緯度、Aの経度、Bの緯度、Bの経度）から直線距離を求めるメソッドを作って計算

・最も簡単には地球を球体とみなして、地球の半径と緯度差から南北距離、経度差から東西距離を出して・・・

・より正確な距離を求める計算式もいろいろあるので実装してみてください

ヒント：Math.toRadians、Math.sqrt

(答え合わせ)

東京駅(35.681143,139.767208)から横浜駅(35.466193,139.622498)の距離 → 27280m

データの格納

ArraylistとHashmap

・データを管理するデータ構造

・データの要素数を事前に定義しなくても良い（拡張可能）

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
:	:

取り出し方

(Arraylist名).get(2) → “アメリカ”

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
:	:

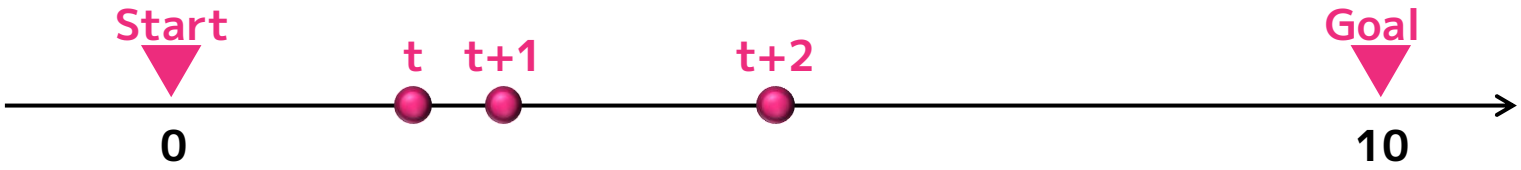
取り出し方

(Hashmap名).get(“Japan”) → “日本”

データの格納

Arraylistの例：1次元でのランダムウォーク

・ゴールまで1ステップに0～1の間でランダムに進む




```
import java.util.ArrayList;                    //ArrayListパッケージのインポートが必要
public class Main {
   public static void main(String[] args) {
      double now = 0.0;                        //現在位置を表す変数
      ArrayList<Double> place = new ArrayList<Double>();
                                    //placeに時系列の位置を格納する、この時点では空っぽ
      while(now < 10){              //現在位置が10を超えるまで計算を繰り返す
         now = now + Math.random();            //現在位置更新、Math.random()は0～1の乱数
         place.add(now);                       //placeに要素を追加する
      }
                                    //この時点でゴールまでの時系列データが書き込まれた
      for (int i = 0; i < place.size(); i++){          //.size()で要素数を取得
         System.out.println(place.get(i));             //コンソールにi番目の要素を表示
      }
      System.out.println("ステップ数:" + place.size());   //要素数(ステップ数)を表示
   }
}
```

課題5

データの集計 – 一致データの探索

・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

・データはArrayListで格納しておく

・ヒント：文字列比較はequals()を使う

input5-1.csv
(ロンドン五輪競技)

```
陸上
水泳
サッカー
テニス
ボート
:
```

input5-2.csv
(競技人口上位30)

```
ウォーキング
ボウリング
水泳
ゴルフ
バドミントン
:
```

output5.csv

```
ウォーキング,0
ボウリング,0
水泳,1
ゴルフ,0
バドミントン,1
:
```

オブジェクト指向

クラスとインスタンス（“学生”をクラスの例で考えると）

・オブジェクトの型を**クラス**として定義する

・ひとつひとつの実体(学生1人1人)が**インスタンス**

・同クラスのインスタンスは共通の性質(学生である)を持つ

・**フィールド**：クラスで共通の状態を表す値

（学籍番号、名前、性別、出身など）

・**メソッド**：クラスで共通の機能

（勉強する、就活する、遊ぶなど）

クラスとインスタンスなどの話は全てできないので、
各自勉強してフォローするように。

オブジェクト指向

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

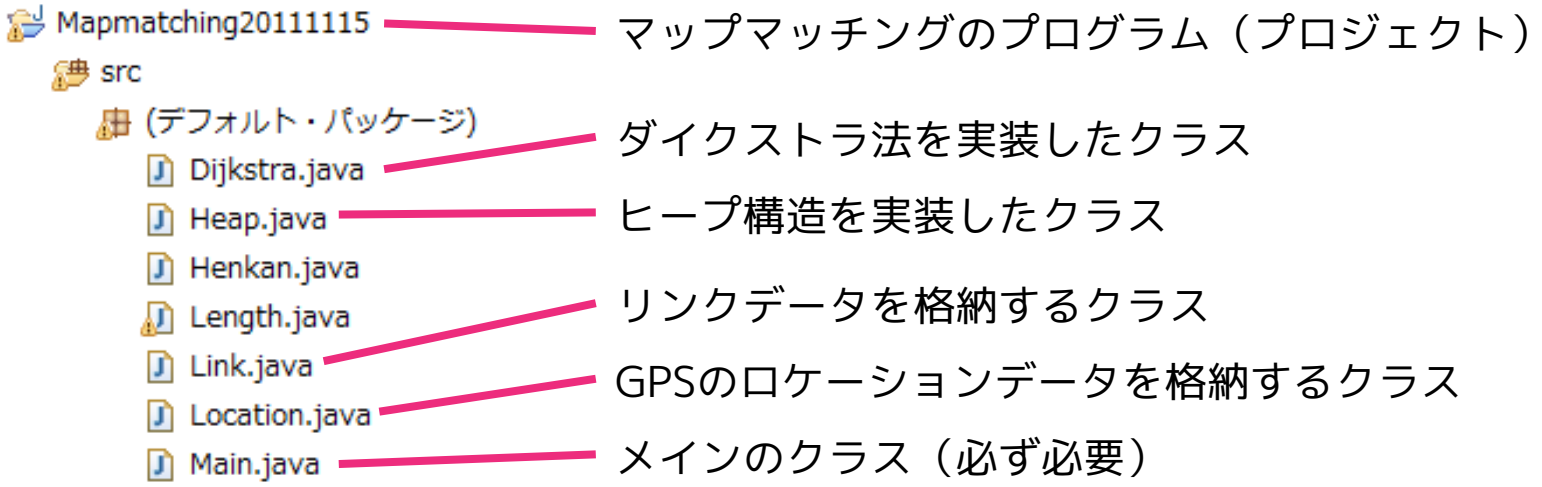
オブジェクト指向

クラスの実装（例として鉄道駅データを扱う）

```
public class Station {              //駅データを格納するクラス
  String name;                      //駅の名前
  double lat;                       //駅の緯度
  double lon;                       //駅の経度
  Station(String n, double x, double y){              //駅の情報を設定する
    name = n;
    lat = x;
    lon = y;
  }
}
```

クラスは「～.java」という一つのファイルになる

例）

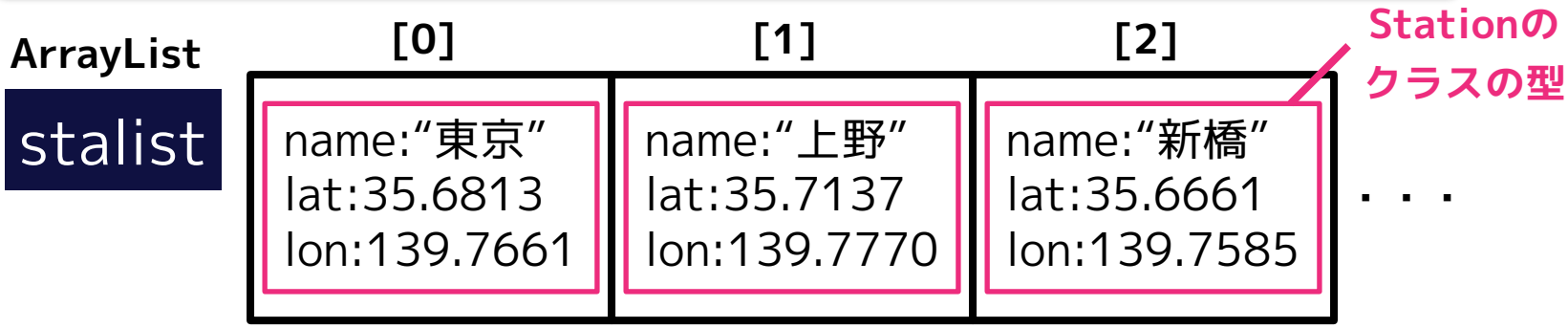


オブジェクト指向

インスタンスの作り方・取り出し方

ここではArrayListの中にStationクラスのインスタンスを加えていく

```
import java.util.ArrayList;                    //ArrayListのインポート
public class Main {
   public static void main(String[] args) {
      ArrayList<Station> stalist = new ArrayList<Station>();       //ArrayList作成
      stalist.add(new Station(“東京”,35.6813,139.7661));
      stalist.add(new Station("上野",35.7137,139.7770));
      stalist.add(new Station("新橋",35.6661,139.7585));
                                   //作ったArrayListのstalistにデータを3つ入れてみた
      for (int i = 0; i < stalist.size(); i++){          //格納データの駅名を全て表示
         System.out.println(stalist.get(i).name);
      }
   }
}
```



課題6

最寄り駅の探索

・input6-2.csvの位置データ1つ1つに対して、最寄り駅を求めて出力する。

・Stationクラスを自分で定義する

・ヒント：課題4のメソッドを使って・・・

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
:
```

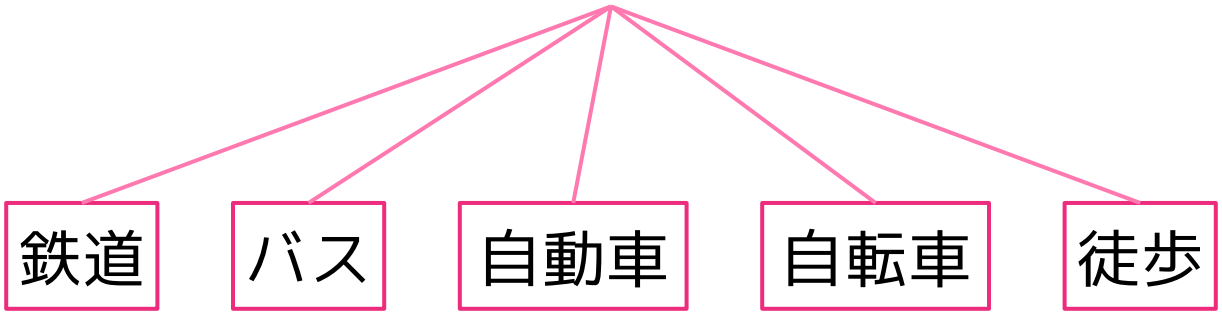
R編

課題7

交通手段選択モデル

・データ(input7.csv)から交通手段選択モデルをつくる
・MNL(多項ロジットモデル)で推定せよ
・効用関数の設定は自分ですること
・推定結果をPowerpoint等で表にまとめてみること

詳細はスタートアップゼミ第2回参照のこと。



乱数の発生

分布と乱数

・Rでは容易に確率分布の操作が可能

d～	確率密度関数
p～	累積確率分布
q～	累積分布の逆関数
r～	乱数の発生

＋

norm	正規分布
unif	一様分布
exp	指数分布
binom	二項分布　など

例)

pnorm(0,0,1)　→　0.5

（平均0、標準偏差1の正規分布の0での累積確率）

runif(5,0,1)　→　0.783　0.456　0.425　0.947　0.884

（0から1の範囲の一様乱数を5つ生成）

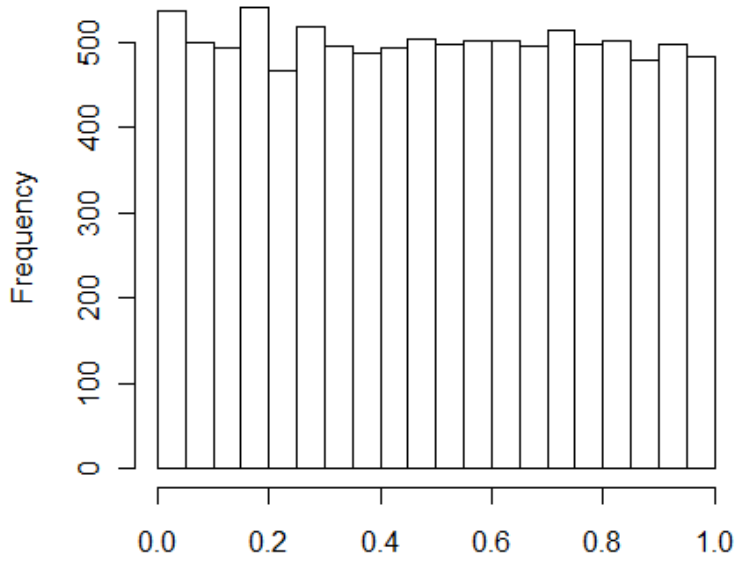
乱数の発生

分布と乱数

・hist()で頻度分布をみる

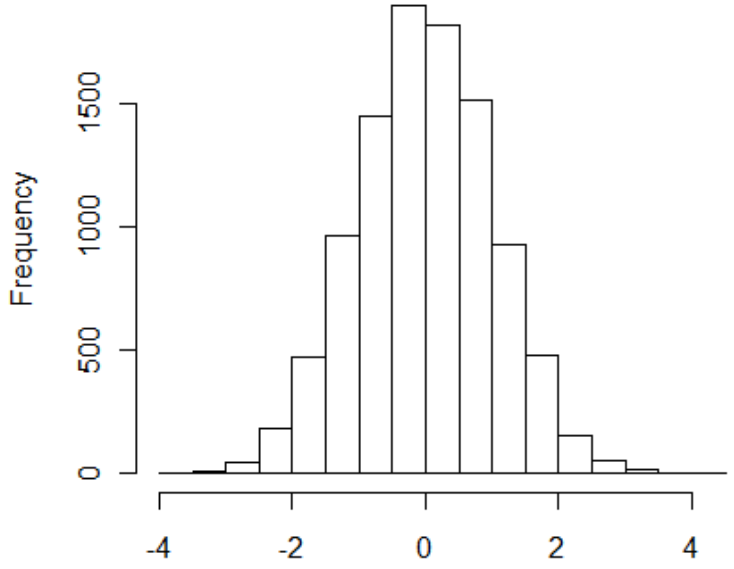
hist(runif(10000,0,1))

0~1の一様乱数を10000個



hist(rnorm(10000,0,1))

平均0、標準偏差1の乱数を10000個



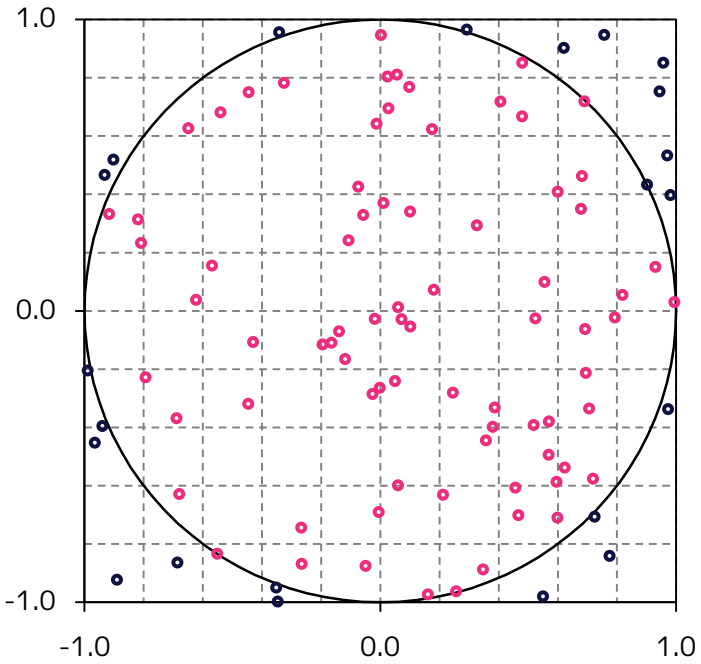
課題8

モンテカルロ法による円周率の計算

・一様分布を発生させて、円の中（中心から1以内）の点の数を数えて、半径1の面積を計算

・乱数を数を変えて精度をみてみる

イメージ



課題9

最適化関数を用いた回帰分析

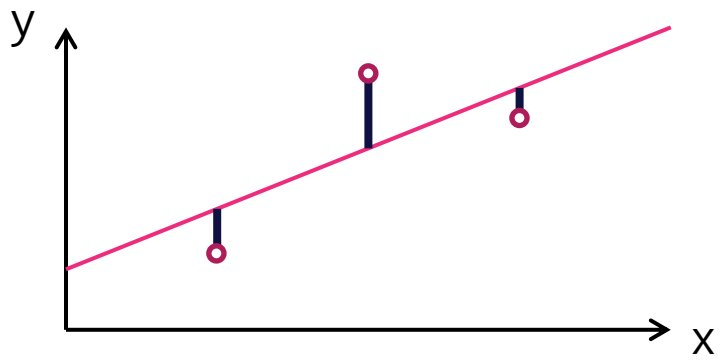
```
x1 <- c(0, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 10)
y1 <- c(0, 0.9, 1.8, 2.7, 3.6, 4.5, 5.4, 6.3, 7.2, 8.1, 9)
Data <- data.frame(X=x1, Y=y1)
plot(Data$X, Data$Y)
```

・上のデータについて回帰分析を行う

・lm関数で`lm(Y~X,data=Data)`とやれば一発だが・・・

・ここでは最小二乗法で計算する（最適化関数を使う）

詳細はスタートアップゼミ第4回参照のこと。



―― の二乗の和が最小になるように関数を定義する