Java　第一回レポート課題

2019/12/21

US162039：梶田悠

2019

目次

[課題内容 2](#_Toc28475448)

[解決手順 2](#_Toc28475449)

[Population\_basicクラス 3](#_Toc28475450)

[void printData() 3](#_Toc28475451)

[void printGraph(String place, Integer population) 3](#_Toc28475452)

[void populationGraph\_all() 3](#_Toc28475453)

[void printTopN(int n, ArrayList<String> plist, HashMap<String, Double> map) 3](#_Toc28475454)

[Population\_type1 4](#_Toc28475455)

[void printData() 4](#_Toc28475456)

[Population\_type2 4](#_Toc28475457)

[void printData() 4](#_Toc28475458)

[使用方法 5](#_Toc28475459)

[実行結果 5](#_Toc28475460)

[考察 5](#_Toc28475461)

[プログラムリスト 6](#_Toc28475462)

[参考文献 6](#_Toc28475463)

[その他 7](#_Toc28475464)

# 課題内容

人口統計のデータをファイル読み込み、都道府県ごとにデータをまとめ、様々な集計を行うプログラムを作成すること。年度によってデータのファイルに記載されている情報が異なるので、それぞれに対応するクラスを定義する必要がある。

各クラスのメンバや関数の仕様などについては配布されたプリントに詳しく記述されているので割愛する。

# 解決手順

配布されたコードと課題内容の要件から変数と関数の宣言は書けてしまうので、実際的には解決手順は設計部分ではなくて各関数の実装部分の処理をどうするかという内容になる。

よって、以下に各関数の機能を満たすための処理について記述する。

## Population\_basicクラス

### void printData()

Population\_basicクラスと派生クラスの持つデータを一覧表示する関数であることが実行例から読み取れた。なのでPopulation\_basicクラスはjinkoMapを全要素出力すれば機能は満たされる。

### void printGraph(String place, Integer population)

引数として地区名と人口を表す値がわたされ、グラフを出力する関数で、人口の100分の1の数だけ“＊”出力していることが実行例とにpopulationData\_UTF8.csvから分かった。なので、単純にpopulationを100で割った結果の整数値回数for文で”\*”を出力すれば機能は満たされる。

### void populationGraph\_all()

この関数は全地区のグラフを出力する関数なので、placeListの全要素を走査して各要素に対応するグラフを出力すれば良い。各要素で前述のprintGrapg()関数を呼び出せば良いことになる。

### void printTopN(int n, ArrayList<String> plist, HashMap<String, Double> map)

課題内容にある様にこの関数は、データのトップn個のデータを表示する関数である。要素数n個のArrayListにソートされた状態になるようにデータのトップn個を詰めて順に表示すれば良い。このトップn個を抽出しArrayListにソートした状態で並べる処理は、mapのデータを走査してArrayListの末尾の要素（最も値が小さい要素）より大きい場合ArrayaListの適切な位置（ソート状態が保たれる位置）にデータを追加する事で実装した。要素数がnを超えた場合末尾の要素を削除する事でn個以上にならないようにした。また要素数が0個の場合は無条件にデータをArrayListに追加、要素数がn個に満たない場合は無条件にArrayListに追加す必要があった。

## Population\_type1

### void printData()

Population\_type1クラスのprintData()は親クラスのprintData()と目的は同じだが保持しているデータが異なるのでその差を吸収する必要がある。Population\_type1クラスはjinkoMap , shuchuMap, densityMap, changeRatioMapを全要素出力するようにオーバーライドすれば良い。

## Population\_type2

### void printData()

Population\_type2クラスも同様に、printData()は親クラスのprintData()と目的は同じだが保持しているデータが異なるのでその差を吸収する必要がある。Population\_type2クラスはjinkoMap , genderRatioMap, changeRatioMapを全要素出力すれば機能は満たされる。

# 使用方法

機能追加はしていないので課題内容に含まれている実行例の通り使えば良いです。

# 実行結果

実行結果 1

|  |
| --- |
| "C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.1\jbr\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.1\lib\idea\_rt.jar=61675:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\User\IdeaProjects\java11\out\production\java11 java11.PopulationStats populationData\_UTF8.csv  データ名：H12年度  都道府県名：北海道 　人口：5683  都道府県名：青森 　人口：1476  都道府県名：岩手 　人口：1416  都道府県名：宮城 　人口：2365  都道府県名：秋田 　人口：1189  都道府県名：山形 　人口：1244  都道府県名：福島 　人口：2127  都道府県名：茨城 　人口：2986  都道府県名：栃木 　人口：2005  都道府県名：群馬 　人口：2025  都道府県名：埼玉 　人口：6938  都道府県名：千葉 　人口：5926  都道府県名：東京 　人口：12064  都道府県名：神奈川 　人口：8490  都道府県名：新潟 　人口：2476  都道府県名：富山 　人口：1121  都道府県名：石川 　人口：1181  都道府県名：福井 　人口：829  都道府県名：山梨 　人口：888  都道府県名：長野 　人口：2215  都道府県名：岐阜 　人口：2108  都道府県名：静岡 　人口：3767  都道府県名：愛知 　人口：7043  都道府県名：三重 　人口：1857  都道府県名：滋賀 　人口：1343  都道府県名：京都 　人口：2644  都道府県名：大阪 　人口：8805  都道府県名：兵庫 　人口：5551  都道府県名：奈良 　人口：1443  都道府県名：和歌山 　人口：1070  都道府県名：鳥取 　人口：613  都道府県名：島根 　人口：762  都道府県名：岡山 　人口：1951  都道府県名：広島 　人口：2879  都道府県名：山口 　人口：1528  都道府県名：徳島 　人口：824  都道府県名：香川 　人口：1023  都道府県名：愛媛 　人口：1493  都道府県名：高知 　人口：814  都道府県名：福岡 　人口：5016  都道府県名：佐賀 　人口：877  都道府県名：長崎 　人口：1517  都道府県名：熊本 　人口：1859  都道府県名：大分 　人口：1221  都道府県名：宮崎 　人口：1170  都道府県名：鹿児島 　人口：1786  都道府県名：沖縄 　人口：1318  データ名：H17年度  都道府県名：北海道 人口：5628　集中地区：4108　人口密度：72.0　増減率：-1.0  都道府県名：青森 人口：1437　集中地区：653　人口密度：150.0　増減率：-2.6  都道府県名：岩手 人口：1385　集中地区：407　人口密度：91.0　増減率：-2.2  都道府県名：宮城 人口：2360　集中地区：1371　人口密度：324.0　増減率：-0.2  都道府県名：秋田 人口：1146　集中地区：386　人口密度：99.0　増減率：-3.7  都道府県名：山形 人口：1216　集中地区：504　人口密度：130.0　増減率：-2.2  都道府県名：福島 人口：2091　集中地区：806　人口密度：152.0　増減率：-1.7  都道府県名：茨城 人口：2975　集中地区：1068　人口密度：488.0　増減率：-0.4  都道府県名：栃木 人口：2017　集中地区：860　人口密度：315.0　増減率：0.6  都道府県名：群馬 人口：2024　集中地区：801　人口密度：318.0　増減率：0.0  都道府県名：埼玉 人口：7054　集中地区：5566　人口密度：1858.0　増減率：1.7  都道府県名：千葉 人口：6056　集中地区：4342　人口密度：1175.0　増減率：2.2  都道府県名：東京 人口：12577　集中地区：12329　人口密度：5751.0　増減率：4.2  都道府県名：神奈川 人口：8792　集中地区：8250　人口密度：3639.0　増減率：3.6  都道府県名：新潟 人口：2431　集中地区：1139　人口密度：193.0　増減率：-1.8  都道府県名：富山 人口：1112　集中地区：398　人口密度：262.0　増減率：-0.8  都道府県名：石川 人口：1174　集中地区：573　人口密度：281.0　増減率：-0.6  都道府県名：福井 人口：822　集中地区：333　人口密度：196.0　増減率：-0.9  都道府県名：山梨 人口：885　集中地区：305　人口密度：198.0　増減率：-0.4  都道府県名：長野 人口：2196　集中地区：764　人口密度：162.0　増減率：-0.8  都道府県名：岐阜 人口：2107　集中地区：822　人口密度：198.0　増減率：-0.1  都道府県名：静岡 人口：3792　集中地区：2216　人口密度：488.0　増減率：0.7  都道府県名：愛知 人口：7255　集中地区：5480　人口密度：1405.0　増減率：3.0  都道府県名：三重 人口：1867　集中地区：762　人口密度：323.0　増減率：0.5  都道府県名：滋賀 人口：1380　集中地区：614　人口密度：344.0　増減率：2.8  都道府県名：京都 人口：2648　集中地区：2162　人口密度：574.0　増減率：0.1  都道府県名：大阪 人口：8817　集中地区：8440　人口密度：4655.0　増減率：0.1  都道府県名：兵庫 人口：5591　集中地区：4198　人口密度：666.0　増減率：0.7  都道府県名：奈良 人口：1421　集中地区：908　人口密度：385.0　増減率：-1.5  都道府県名：和歌山 人口：1036　集中地区：408　人口密度：219.0　増減率：-3.2  都道府県名：鳥取 人口：607　集中地区：207　人口密度：173.0　増減率：-1.0  都道府県名：島根 人口：742　集中地区：179　人口密度：111.0　増減率：-2.5  都道府県名：岡山 人口：1957　集中地区：856　人口密度：275.0　増減率：0.3  都道府県名：広島 人口：2877　集中地区：1832　人口密度：339.0　増減率：-0.1  都道府県名：山口 人口：1493　集中地区：708　人口密度：244.0　増減率：-2.3  都道府県名：徳島 人口：810　集中地区：260　人口密度：195.0　増減率：-1.7  都道府県名：香川 人口：1012　集中地区：330　人口密度：540.0　増減率：-1.0  都道府県名：愛媛 人口：1468　集中地区：748　人口密度：259.0　増減率：-1.7  都道府県名：高知 人口：796　集中地区：341　人口密度：112.0　増減率：-2.2  都道府県名：福岡 人口：5050　集中地区：3525　人口密度：1015.0　増減率：0.7  都道府県名：佐賀 人口：866　集中地区：246　人口密度：355.0　増減率：-1.2  都道府県名：長崎 人口：1479　集中地区：688　人口密度：361.0　増減率：-2.5  都道府県名：熊本 人口：1842　集中地区：805　人口密度：249.0　増減率：-0.9  都道府県名：大分 人口：1210　集中地区：537　人口密度：191.0　増減率：-0.9  都道府県名：宮崎 人口：1153　集中地区：512　人口密度：149.0　増減率：-1.4  都道府県名：鹿児島 人口：1753　集中地区：689　人口密度：191.0　増減率：-1.8  都道府県名：沖縄 人口：1362　集中地区：893　人口密度：599.0　増減率：3.3  データ名：H21年度  都道府県名：北海道 人口：5507　人口性比：89.4　増減率：-5.2  都道府県名：青森 人口：1379　人口性比：88.6　増減率：-9.4  都道府県名：岩手 人口：1340　人口性比：91.1　増減率：-8.7  都道府県名：宮城 人口：2336　人口性比：94.0　増減率：-2.0  都道府県名：秋田 人口：1096　人口性比：88.5　増減率：-11.0  都道府県名：山形 人口：1179　人口性比：92.1　増減率：-8.1  都道府県名：福島 人口：2040　人口性比：94.2　増減率：-6.2  都道府県名：茨城 人口：2960　人口性比：99.0　増減率：-1.2  都道府県名：栃木 人口：2006　人口性比：98.8　増減率：-2.7  都道府県名：群馬 人口：2007　人口性比：96.9　増減率：-2.6  都道府県名：埼玉 人口：7130　人口性比：101.2　増減率：2.4  都道府県名：千葉 人口：6139　人口性比：99.8　増減率：2.8  都道府県名：東京 人口：12868　人口性比：99.0　増減率：2.3  都道府県名：神奈川 人口：8943　人口性比：101.6　増減率：2.9  都道府県名：新潟 人口：2378　人口性比：93.6　増減率：-5.4  都道府県名：富山 人口：1095　人口性比：92.9　増減率：-5.4  都道府県名：石川 人口：1165　人口性比：93.2　増減率：-2.6  都道府県名：福井 人口：808　人口性比：93.5　増減率：-5.7  都道府県名：山梨 人口：867　人口性比：95.7　増減率：-4.3  都道府県名：長野 人口：2159　人口性比：94.5　増減率：-5.3  都道府県名：岐阜 人口：2092　人口性比：93.9　増減率：-4.1  都道府県名：静岡 人口：3792　人口性比：97.3　増減率：-2.1  都道府県名：愛知 人口：7418　人口性比：101.1　増減率：1.9  都道府県名：三重 人口：1870　人口性比：94.7　増減率：-3.0  都道府県名：滋賀 人口：1405　人口性比：97.8　増減率：2.2  都道府県名：京都 人口：2622　人口性比：92.2　増減率：-2.6  都道府県名：大阪 人口：8801　人口性比：93.7　増減率：-0.6  都道府県名：兵庫 人口：5583　人口性比：91.7　増減率：-0.6  都道府県名：奈良 人口：1399　人口性比：90.0　増減率：-3.4  都道府県名：和歌山 人口：1004　人口性比：88.4　増減率：-7.0  都道府県名：鳥取 人口：591　人口性比：91.1　増減率：-7.7  都道府県名：島根 人口：718　人口性比：90.8　増減率：-9.3  都道府県名：岡山 人口：1942　人口性比：92.1　増減率：-3.4  都道府県名：広島 人口：2863　人口性比：93.5　増減率：-2.0  都道府県名：山口 人口：1455　人口性比：89.0　増減率：-5.7  都道府県名：徳島 人口：789　人口性比：90.2　増減率：-6.2  都道府県名：香川 人口：999　人口性比：92.3　増減率：-3.4  都道府県名：愛媛 人口：1436　人口性比：88.8　増減率：-5.5  都道府県名：高知 人口：766　人口性比：88.0　増減率：-9.0  都道府県名：福岡 人口：5053　人口性比：89.7　増減率：-0.2  都道府県名：佐賀 人口：852　人口性比：88.6　増減率：-4.4  都道府県名：長崎 人口：1430　人口性比：87.3　増減率：-7.2  都道府県名：熊本 人口：1814　人口性比：88.7　増減率：-4.0  都道府県名：大分 人口：1195　人口性比：89.1　増減率：-4.6  都道府県名：宮崎 人口：1132　人口性比：88.4　増減率：-4.1  都道府県名：鹿児島 人口：1708　人口性比：87.3　増減率：-5.7  都道府県名：沖縄 人口：1382　人口性比：95.9　増減率：4.5  =====H12 年度の全都道府県の人口グラフ=====  都道府県名：北海道 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：青森 \*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：岩手 \*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：宮城 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：秋田 \*\*\*\*\*\*\*\*\*\*\*  都道府県名：山形 \*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：福島 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：茨城 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：栃木 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：群馬 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：埼玉 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：千葉 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：東京 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：神奈川 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：新潟 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：富山 \*\*\*\*\*\*\*\*\*\*\*  都道府県名：石川 \*\*\*\*\*\*\*\*\*\*\*  都道府県名：福井 \*\*\*\*\*\*\*\*  都道府県名：山梨 \*\*\*\*\*\*\*\*  都道府県名：長野 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：岐阜 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：静岡 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：愛知 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：三重 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：滋賀 \*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：京都 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：大阪 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：兵庫 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：奈良 \*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：和歌山 \*\*\*\*\*\*\*\*\*\*  都道府県名：鳥取 \*\*\*\*\*\*  都道府県名：島根 \*\*\*\*\*\*\*  都道府県名：岡山 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：広島 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：山口 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：徳島 \*\*\*\*\*\*\*\*  都道府県名：香川 \*\*\*\*\*\*\*\*\*\*  都道府県名：愛媛 \*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：高知 \*\*\*\*\*\*\*\*  都道府県名：福岡 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：佐賀 \*\*\*\*\*\*\*\*  都道府県名：長崎 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：熊本 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：大分 \*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：宮崎 \*\*\*\*\*\*\*\*\*\*\*  都道府県名：鹿児島 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県名：沖縄 \*\*\*\*\*\*\*\*\*\*\*\*\*  =====H17 年度人口増加率トップ10=====  都道府県名：東京 4.2  都道府県名：神奈川 3.6  都道府県名：沖縄 3.3  都道府県名：愛知 3.0  都道府県名：滋賀 2.8  都道府県名：千葉 2.2  都道府県名：埼玉 1.7  都道府県名：静岡 0.7  都道府県名：兵庫 0.7  都道府県名：福岡 0.7  =====H21 年度人口増加率トップ10=====  都道府県名：沖縄 4.5  都道府県名：神奈川 2.9  都道府県名：千葉 2.8  都道府県名：埼玉 2.4  都道府県名：東京 2.3  都道府県名：滋賀 2.2  都道府県名：愛知 1.9  都道府県名：福岡 -0.2  都道府県名：大阪 -0.6  都道府県名：兵庫 -0.6  人口分析ツール  都道府県を入力->  北海道  =====北海道の人口グラフ=====  H12 年度:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  H17 年度:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  H21 年度:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県を入力->  埼玉  =====埼玉の人口グラフ=====  H12 年度:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  H17 年度:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  H21 年度:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  都道府県を入力->  鳥取  =====鳥取の人口グラフ=====  H12 年度:\*\*\*\*\*\*  H17 年度:\*\*\*\*\*\*  H21 年度:\*\*\*\*\*  都道府県を入力-> |

# 考察

## printGraph関数の引数の冗長性

ポータルで配布されたコードではprintGraph()関数の引数は地区名を表すString型変数と、その地区の人口を表す整数値の二つだったが。この関数は人口に応じて”\*”を出力する関数なので地区名を引数として受け取る必要性がない。JinkoMapから人口を取得する為に地区名のString変数を利用するのであれば、人口を表す整数値の方が必要なくなる。よってどちらかを削除するべきと考えられる。地区名のString変数を受け取る方が妥当だと考えた。理由は呼び出し側でjinkoMapから人口データを取得する記述をしなくてよくなるのでコードがシンプルになって読み易いことと、逆に人口を表す整数値を引数として受け取る場合は単純に整数に応じて”\*”を出力するだけの関数になるのでPopulation\_○○クラスが担う役割としては適切ではないように考えるからである。視覚的に見やすい出力をまとめるようにライブラリに含まれていた方が意味的に適切に思える。

## GUIの採用

今回のプログラムは最終的にデータを見やすく出力するものなので、コンソール出力よりもswingなどを用いてGUIで表現するほうが適切である。Swingによる視覚化は実装していないが、ウィンドウを分割して複数のグラフを同時に出力することもできるし表示するデータを条件で絞るような機能を実装する場合に条件指定のインターフェースをコンソールを使うより分かりやすく使いやすくなることも期待できる。

## エラーの出力

配布されたコードの人口分析ツールの部分を見ると、都道府県の入力をreadLine()で取得するtry-catch構文でcatchのブロックに何も記述されていなかった。もしエラーが起きてcatchブロックが実行された場合にユーザーからみると何も起きずにインタラクションが無いので状況を把握できなくなる。今回のプログラムでは問題なさそうだがエラー出力をしないとデバッグ時にもバグを追いづらくなる。なので念の為System.out.println(e);を追記した。

# プログラムリスト

GithubにintelliJのプロジェクトフォルダごとアップロードしている為適宜以下のリンクを参照して下さい。

<https://github.com/HarukaKajita/java_report1>

コード 1 PopulationStats.java

|  |
| --- |
| package java11;  import java.io.BufferedReader;  import java.io.FileNotFoundException;  import java.io.FileReader;  import java.io.IOException;  import java.io.InputStreamReader;  public class PopulationStats {  Population\_basic h12 = new Population\_basic ();  Population\_type1 h17 = new Population\_type1();  Population\_type2 h21 = new Population\_type2();  void readFile(String filename){  try {  BufferedReader reader = new BufferedReader(new FileReader(filename));  String line;  h12.name="H12 年度";  h17.name="H17 年度";  h21.name="H21 年度";  reader.readLine();  while ((line=reader.readLine())!=null) {  String[] items = line.split(",");  h12.placeList.add(items[0]);  h17.placeList.add(items[0]);  h21.placeList.add(items[0]);  h12.jinkoMap.put(items[0],Integer.parseInt(items[1]));  h17.jinkoMap.put(items[0], Integer.parseInt(items[2]));  h17.shuchuMap.put(items[0], Integer.parseInt(items[3]));  h17.densityMap.put(items[0], Double.parseDouble(items[4]));  h17.changeRatioMap.put(items[0], Double.parseDouble(items[5]));  h21.jinkoMap.put(items[0], Integer.parseInt(items[6]));  h21.genderRatioMap.put(items[0], Double.parseDouble(items[7]));  h21.changeRatioMap.put(items[0], Double.parseDouble(items[8]));  }  } catch (FileNotFoundException e) {  System.out.println("FileReaderから投げられた例外"+e);  } catch (IOException e) {  System.out.println("readLineから投げられた例外"+e);  }  }  public static void main(String[] args) {  PopulationStats ps = new PopulationStats();  //入力ファイルを読み込み、年度別に格納  ps.readFile(args[0]);  ps.h12.printData();  ps.h17.printData();  ps.h21.printData();  System.out.println("=====H12 年度の全都道府県の人口グラフ=====");  //100,000人を\*として棒グラフで人口を表現  ps.h12.populationGraph\_all();  System.out.println("=====H17 年度人口増加率トップ10=====");  ps.h17.printTopN(10,ps.h17.placeList,ps.h17.changeRatioMap);  System.out.println("=====H21 年度人口増加率トップ10=====");  ps.h21.printTopN(10,ps.h21.placeList,ps.h21.changeRatioMap);  System.out.println("人口分析ツール");  BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));  while(true){  try {  System.out.println("都道府県を入力->");  String place = reader.readLine();  ps.printJinko(place);  }catch (Exception e){  //例外処理をするべき  System.out.println(e);  }  }  }  void printJinko(String place) {  System.out.println("====="+place+"の人口グラフ=====");  System.out.print("H12 年度:");  h12.printGraph(place);  System.out.print("H17 年度:");  h17.printGraph(place);  System.out.print("H21 年度:");  h21.printGraph(place);  }  } |

コード 2 Population\_basic.java

|  |
| --- |
| package java11;  import java.nio.charset.Charset;  import java.util.ArrayList;  import java.util.HashMap;  interface PopulationAnalyzer{  void printData();  void populationGraph\_all();  }  public class Population\_basic implements PopulationAnalyzer{  String name = "";  ArrayList<String> placeList = new ArrayList<>();  HashMap<String, Integer> jinkoMap = new HashMap<>();  Population\_basic(){  }  public void printTopN(int n, ArrayList<String> plist, HashMap<String, Double> map){  ArrayList<Integer> topIndexes = new ArrayList<Integer>();  //トップn個を降順に並べたリストを作る  for (int i = 0; i < plist.size(); i++) {  if(i == 0) {  topIndexes.add(i);  continue;  }  double bottomOfTop = map.get(plist.get(topIndexes.size()-1));  double ration = map.get(plist.get(i));  if(bottomOfTop < ration || topIndexes.size() < n){  //降順になるような適切な位置に挿入  for (int j = 0; j < topIndexes.size(); j++){  int index = topIndexes.get(j);  String p = plist.get(index);  double r = map.get(p);  if(r < ration){  topIndexes.add(j, i);  break;  }else if(j == topIndexes.size()-1){  //最後尾に挿入  topIndexes.add(i);  break;  }  }  //不要な要素を削除  if(topIndexes.size() > n)topIndexes.remove(n);  }  }  //出力  for (int index : topIndexes) {  String place = plist.get(index);  String placeStr = getPrefixStr(place);  System.out.print(placeStr);  Double value = map.get(place);  System.out.println(value);  }  }  //レポートの課題内容で使用定義されていなかったが配布されたコード内で使用していたので実装  //単純にデータを一覧出力  @Override  public void printData() {  //H12向けの処理。他の年度ではオーバーライドして実装。  System.out.println("データ名：H12年度");  for (String place : placeList) {  String placeStr = getPrefixStr(place);  System.out.print(placeStr + "　");  System.out.println("人口：" + jinkoMap.get(place));  }  }  public void printGraph(String place) {  // TODO:文字列整形 https://qiita.com/Lilly008000/items/00876d8c61ce36bd5fba  //100人につき"\*"を1つ出力するグラフ  int scaledPopulation = jinkoMap.get(place) / 100;  for (int i = 0; i < scaledPopulation; i++) System.out.print("\*");  System.out.println("");  }  @Override  public void populationGraph\_all() {  for (String place: placeList) {  // TODO:文字列整形 https://qiita.com/Lilly008000/items/00876d8c61ce36bd5fba  //文字列整形  String placeStr = getPrefixStr(place);  System.out.print(placeStr);  //100人につき"\*"を1つ出力するグラフ  printGraph(place);  }  }  //"都道府県名：○○\_\_："になるように文字列整形  protected String getPrefixStr(String place){  String placeStr = format(place, 10);  placeStr = "都道府県名：" + placeStr;  return placeStr;  }  protected String format(String target, int length){  int byteDiff = (getByteLength(target, Charset.forName("UTF-8"))-target.length())/2;  return String.format("%-"+(length-byteDiff)+"s", target);  }  protected int getByteLength(String string, Charset charset) {  return string.getBytes(charset).length;  }  } |

コード 3 Population\_type1.java

|  |
| --- |
| package java11;  import java.util.HashMap;  public class Population\_type1 extends Population\_basic {  HashMap<String, Integer> shuchuMap = new HashMap<>();  HashMap<String, Double> densityMap = new HashMap<>();  HashMap<String, Double> changeRatioMap = new HashMap<>();  Population\_type1(){  }  @Override  public void printData() {  //H17向けの処理。他の年度ではオーバーライドして実装。  System.out.println("データ名：H17年度");  for (String place : placeList) {  String placeStr = getPrefixStr(place);  System.out.print(placeStr);  System.out.print("人口：" + jinkoMap.get(place) + "　");  System.out.print("集中地区：" + shuchuMap.get(place) + "　");  System.out.print("人口密度：" + densityMap.get(place) + "　");  System.out.print("増減率：" + changeRatioMap.get(place) + "　");  System.out.println("");  }  }  } |

コード 4 Population\_type2.java

|  |
| --- |
| package java11;  import java.util.HashMap;  public class Population\_type2 extends Population\_basic{  HashMap<String, Double> genderRatioMap = new HashMap<>();  HashMap<String, Double> changeRatioMap = new HashMap<>();  Population\_type2(){  }  @Override  public void printData() {  //H21向けの処理。他の年度ではオーバーライドして実装。  System.out.println("データ名：H21年度");  for (String place : placeList) {  String placeStr = getPrefixStr(place);  System.out.print(placeStr);  System.out.print("人口：" + jinkoMap.get(place) + "　");  System.out.print("人口性比：" + genderRatioMap.get(place) + "　");  System.out.print("増減率：" + changeRatioMap.get(place) + "　");  System.out.println("");  }  }  } |

# 参考文献

<https://qiita.com/Lilly008000/items/00876d8c61ce36bd5fba>

<https://www.sejuku.net/blog/14886>

<https://java-reference.com/java_collection_arraylist.html>