15\_基本情報アルゴリズム①

**課題1**

ユークリッドの互除法により最大公約数を求めて表示するプログラムを作成しなさい。

また、整数以外が入力された場合の例外処理も実装すること。

※ユークリッドの互除法とは

2つの整数の最大公約数を求めるアルゴリズム。

割り切れるまであまりで互いに割り続けるという手法で、まず2つの整数を割る。

そのあまりで割った数を割り、さらにあまりで割った数を割る…を割り切れるまで続ける。

そして商が0になった時の割った数が最大公約数となる。

ファイル名：Kad15\_1.java

＜実行結果＞

整数1＞1649

整数2＞221

最大公約数：17

＜実行結果＞

整数1＞64

整数2＞abc

入力値が不正です。

＜実行結果＞

整数1＞35

整数2＞14

最大公約数：7

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | int | euclid | int x, int y |

◆euclidメソッドの処理

引数xとyの最大公約数をユークリッドの互除法に基づいて求め、戻り値として返す。

int rem; //あまり

//yが0でない間ループ

while(～省略～){

～省略～; //xをyで割ったあまりをremに代入

～省略～; //xをyで更新

～省略～; //yをあまりで更新

}

～省略～; //x（最大公約数）を返す

◆mainメソッドの処理

①2つの整数を入力し、int型変数num1とnum2に代入する。

この際、整数以外が入力された場合の例外処理をtry-catch文で行い、以降の処理は行わない。

②2つの入力値を引数にeuclidメソッドを呼び出し、戻り値の最大公約数を表示する。

**課題2**

エラトステネスの篩により素数を求めて表示するプログラムを作成しなさい。

また、整数以外が入力された場合の例外処理も実装すること。

※エラトステネスの篩とは

指定された整数n以下の素数を発見するアルゴリズム。

ある整数nが2からn-1まで、いずれの整数でも割り切れなければ整数nは素数となる。

例えば素数13は、2 3 4…のどの数でも割り切ることはできないので素数だと求められる。

ファイル：Kad15\_2.java

＜実行結果＞

いくつまでの素数を求めますか？＞abc

入力値が不正です。

＜実行結果＞

いくつまでの素数を求めますか？＞25

2 3 5 7 11 13 17 19 23

＜実行結果＞

いくつまでの素数を求めますか？＞100

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | boolean[] | eratosthenes | int n |

◆eratosthenesメソッドの処理

引数で受け取った整数n以下の素数をエラトステネスの篩を使用して求めて、boolean型配列

素数フラグに結果を代入し、配列ごと戻り値として返す。

①素数フラグ配列を宣言する(true:素数, false:非素数)

boolean[] flag = new boolean[～省略～];

②素数フラグ配列を初期化する。

0と1は素数になり得ないので、2からnまでの値に素数候補としてtrueを代入する。

for(int i = 2; i <= n; i++){

flag[i] = true;

}

↑このように記述しても良いが、同じ値をセットできるArraysクラスのfillメソッドが存在する。

import java.util.Arrays; //Arraysクラスをインポート

Arrays.fill(flag, 2, n, true);

↑このように記述すればfor文を使用せずに同じ処理を1行で記述できる。

それぞれの引数の内容は以下の通りである。

Arrays.fill(値を変更したい配列, 開始要素番号, 終了要素番号, セットする値);

③エラトステネスの篩に基づいて素数を求め、素数ではない数の素数フラグをfalseに変更する。

ループの条件にはMath.sqrtメソッドを使用すると良い。

※Math.sqrt(n)とは

nの平方根を返すメソッド。

平方根とはある数に対して、平方すると元の値に等しくなる数のこと。

具体的には、A × A = n のAを指す。

例えば、n=100とすると平方根は10となり、n=25とすると平方根は5となる。

//素数の倍数を素数候補から除外する(非素数をfalseに変更)

for(int i = 2; ～省略～; i++){

for(int j = i \* 2; j <= n; ～省略～){

flag[j] = false;

}

}

④素数フラグ配列を戻り値として返す。

◆mainメソッドの処理

①いくらまでの素数を求めるか、整数を入力しint型変数inNumに代入する。

この際、整数以外が入力された場合の例外処理をtry-catch文で行い、以降の処理は行わない。

②boolean型素数フラグ配列primeNumFlagを宣言し、eratosthenesメソッドを呼び出した戻り値で初期化する。

// true:素数, false:非素数

boolean[] primeNumFlag = eratosthenes(～省略～);

③求めた素数フラグの値を出力する。

0と1は素数になり得ないので、2からinNumまでの値がtrueの場合、素数として表示する。

//求めた素数を表示

for(int i = 2; ～省略～; i++){

if(～省略～){

System.out.print(i + " ");

}

}