Ex\_総合演習⑥

**課題1**

0~9の数値を重複させずにn桁個の数列を生成し表示するプログラムを作成しなさい。

nの許容桁数は1~10とし、今課題ではn = 4とすること。

また、課題2の為に生成した数値を配列（answer）の各要素に1つずつ代入して値を保持すること。

ファイル名：Kad26\_1.java

◆n = 1の場合 ◆n = 10の場合

＜実行結果＞

7

＜実行結果＞

3469

＜実行結果＞

0281394567

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | void | shuffle | int[] data |

◆shuffleメソッドの処理

引数で受け取った配列の値をシャッフルする。

（Kad23\_1で作成した同じメソッドをコピーし流用する）

◆mainメソッドの処理

①変数・配列宣言を行う。

final int DIGIT = 4; //桁数(1~10まで、今課題は4とする)

int[] answer = new int[DIGIT]; //解答配列

int[] number = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; //元の数値配列

②shuffleメソッドを呼び出し、元となる数値配列をシャッフルする。

③シャッフルした配列の先頭要素から順に桁数個値を取り出し、解答配列に代入する。

④解答配列を表示する。

※完成したらDIGITの値を変更して、実行結果のように桁数が流動するか確かめなさい。

**課題2**

課題1を元に、マスターマインドを行うプログラムを作成しなさい。

※マスターマインドとは、0~9の中で重複していないn桁の数値を当てるゲームのこと。

数値を答えると、その数値がどこまで一致しているかどうかの結果を返し、その情報を頼りに数値を

絞っていく。返す情報は次の２つ。

EAT … 答えた数値の位置も数値も合っている数

BITE … 答えた数値の位置は違うが、数値は合っている数

例えば、解答が「2374」の時に「1234」と答えた場合、4が位置も数値も合っているので1EAT、

2と3が位置は違うが、数値は合っているので2BITEとなる。

◆追加の仕様

①0を扱う為、答える入力値は文字として取得し、数値に変換して解答と値を比較する。

②解答は数値が重複しないが、入力の際に重複させるのは可とする（111や7979と答えても良い）

③入力値の桁チェックを行い、指定した桁ではない場合は不正とし再入力させる。

④入力値が整数に変換できない場合、例外処理行い再入力させる。

⑤入力値が整数の場合、EAT・BITE判定を行い結果と答えた回数を表示する。

⑥指定した桁数分EATになった（解答と完全一致した）場合、ゲームクリアとして終了する。

ファイル名：Kad26\_2.java

◆3桁の場合

＜実行結果＞

マスターマインド！3桁の数値を当てよう

数値を当ててください＞a

桁数が違います。

数値を当ててください＞abc

入力値が不正です。

数値を当ててください＞012

1回目：1EAT 0BITE

数値を当ててください＞345

2回目：1EAT 0BITE

数値を当ててください＞678

3回目：0EAT 0BITE

数値を当ててください＞049

4回目：1EAT 1BITE

数値を当ててください＞942

5回目：3EAT 0BITE

CLEAR!!

＜実行結果＞

マスターマインド！4桁の数値を当てよう

数値を当ててください＞0123

1回目：3EAT 0BITE

数値を当ててください＞4567

2回目：0EAT 0BITE

数値を当ててください＞8456

3回目：0EAT 0BITE

数値を当ててください＞0129

4回目：2EAT 1BITE

数値を当ててください＞9123

5回目：2EAT 1BITE

数値を当ててください＞0923

6回目：2EAT 1BITE

数値を当ててください＞0193

7回目：4EAT 0BITE

CLEAR!!

　　◆マスターマインドの考え方解説

＜実行結果＞

マスターマインド！4桁の数値を当てよう

数値を当ててください＞0123

1回目：0EAT 2BITE

数値を当ててください＞4567

2回目：0EAT 1BITE

数値を当ててください＞6789

3回目：0EAT 1BITE

数値を当ててください＞1295

4回目：2EAT 0BITE

数値を当ててください＞1248

5回目：4EAT 0BITE

CLEAR!!

0123の内2つ使用されていることが分かった。

4567の内1つ使用されているので、残りの8 or 9の

どちらかが使用されていることが分かった。

8 or 9の使用が確定しているのに1BITEしかしていないので、6と7は使用されていないことが分かる。

この時点で0123から2つ、45から1つ、89から1つ

使用されていることが分かった。

0123から12を候補に、45から5を候補に、

89から9を候補に位置を変えると2EATとなった。

この時点で使用数値は1248 or 0395だと分かる。

4回目の結果から12がEATしていると仮定すれば、1248 or 1284だが3回目の結果が0EATなので1284はあり得ない。よって1248が候補となる。

95がEATしていると仮定すれば0395 or 3095だが、

1回目の結果が0EATなので0395はあり得ない。

よって3095が候補となり、1248 or 3095の二択に

絞られることが分かった。

二択の勝負に勝った。

◆mainメソッドの処理

①以下の変数・配列を追加する。

int count = 0; //答えた回数のカウンタ

int eat = 0; //位置も数値も合っている数

int bite = 0; //位置は違うが、数値は合っている数

String inStr; //入力文字代入変数

int[] inNum = new int[DIGIT]; //入力値を1桁ずつ分解し代入する配列

②EATの数が桁数と違う間、以降の処理をループする。

③入力値をnext()メソッドで文字として受け取り、変数に代入する。

④入力文字の桁数チェックを行う。入力された文字の長さが桁数と違う場合、ループ開始時に戻る。

⑤入力文字を先頭から1字ずつ分解し、数値に変換させて配列に代入する。

　この際、整数に変換できない場合の例外処理もtry-catch文で記述する。

⑥1字ずつEAT・BITE判定を行い、ヒットした場合それぞれのカウンタをインクリメントする。

⑦答えた回数のカウンタをインクリメントし、結果を表示する。