

情報処理技法 (Javaプログラミング)₂

第6回 第1回課題フィードバック

人間科学科コミュニケーション専攻
白銀 純子

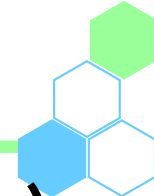


第6回の内容



- ◆ プログラミングでの割り算の使い方
- ◆ フラグ

前回の出席課題の回答

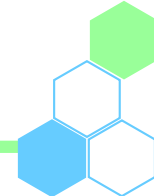


◆ 「クラス」と「オブジェクト」の違いについて、具体例を交えて説明しなさい。

解答例:

「クラス」とは、プログラム中で扱う様々なもの(現実世界の実物)を分類したものであり、「オブジェクト」とは、クラスにあてはまる具体的な実物である。具体的には、図書館の蔵書管理システムを考えると、システムで扱うデータとして、本のデータがある。「本」が「クラス」に相当する。これは、「本」にあてはまる実物は図書館の中にたくさん存在するためである。また、「蔵書ID: 0001, 基礎講座 Java, 白銀純子, 毎日コミュニケーションズ, 2010」のように、具体的にただ1つに特定できる実物がオブジェクトである。

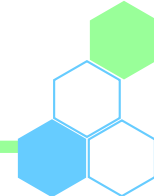
課題のフィードバック～割り算～(1)



◆ int型での割り算の使い方

- ◆ int型の割り算は、商と余りを計算するテクニックが便利
- ◆ ある数から何かを取り除き、その取り除いた数からまたさらに取り除く、ということを繰り返す場合に有効
 - ◆ 引き算ではなく割り算で取り除く場合

課題のフィードバック～割り算～(2)



Ex. 第1回課題のその2: 数を各桁に分解するには?

- 分解したい数をinput、各桁を入れる配列をnumとする
- 変数inputの値が123だとすると...

for文で、inputの値を10で割った商と余りを計算する

- iが0のときのinputに対する計算: 123 → 商: 12, 余り: 3
 - ✓ 余りを配列num[0]に入れ、商でinputの値を更新
- iが1のとき: 12 → 商: 1, 余り: 2
 - ✓ 余りを配列num[1]に入れ、商でinputの値を更新
- iが2のとき: 1 → 商: 0, 余り: 1
 - ✓ 余りを配列num[2]に入れ、商でinputの値を更新
- iが3のとき: 0 → 商: 0, 余り: 0 になるので、終了

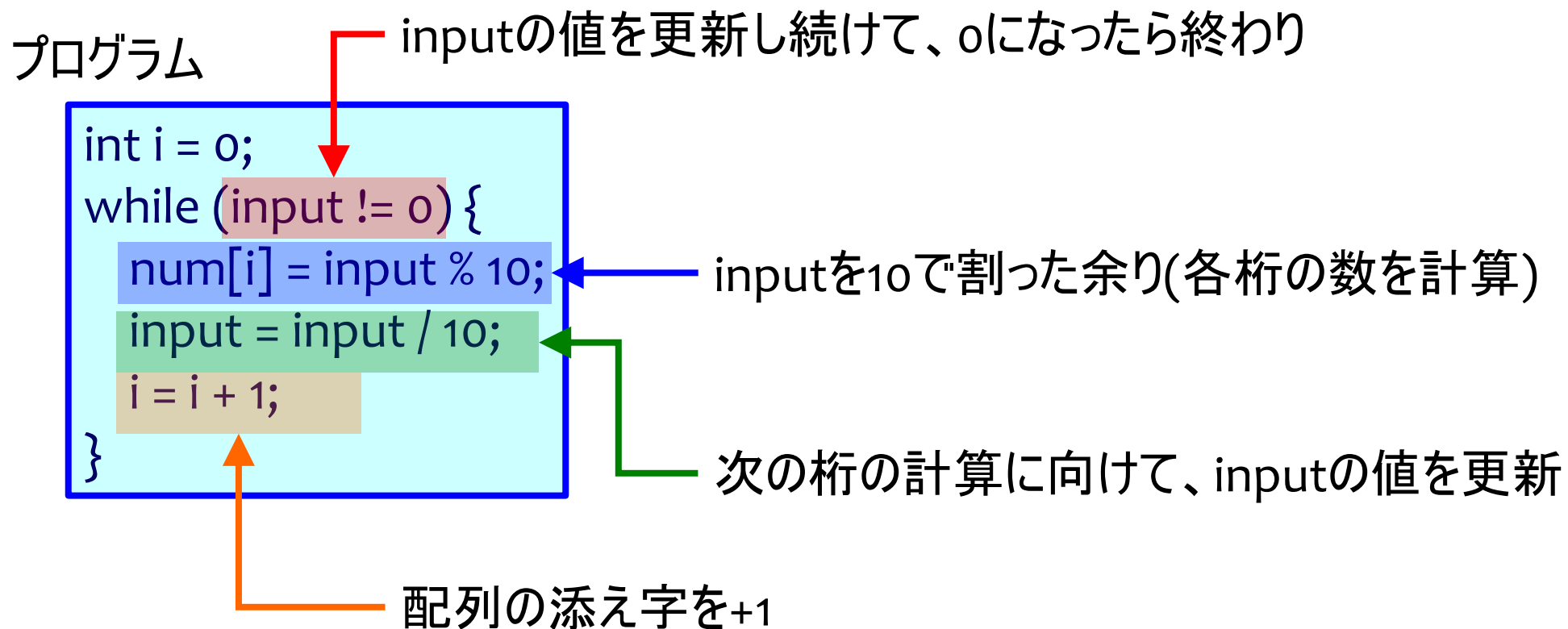
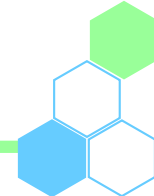


: 商



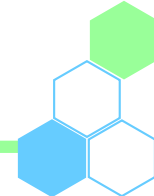
: 余り

課題のフィードバック～割り算～(3)



※標準入力された数を、文字列の段階で数えた文字数がそのまま桁数になるので、その桁数でfor文で分解してもOK

課題のフィードバック～割り算～(4)

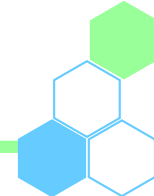


Ex. 第1回課題のその3: 硬貨の枚数を計算するには?

- 枚数を計算したい金額をmoneyとする
- 変数moneyの値が987だとすると...

moneyの値を硬貨の金額で割った商と余りを計算する

- 500円玉の枚数計算: 987 → 商: 1, 余り: 487
 - ✓ 商が500円玉の枚数、余りでmoneyの値を更新
- 100円玉の枚数計算: 487 → 商: 4, 余り: 87
 - ✓ 商が100円玉の枚数、余りでmoneyの値を更新
- 50円玉の枚数計算: 87 → 商: 1, 余り: 37
 - ✓ 商が50円玉の枚数、余りでmoneyの値を更新
- 10円玉の枚数計算: 37 → 商: 3, 余り: 7
 - ✓ 商が10円玉の枚数、余りでmoneyの値を更新
- 5円玉の枚数計算: 7 → 商: 1, 余り: 2
 - ✓ 商が10円玉の枚数、余りが1円玉の枚数



課題のフィードバック～割り算～(4)

硬貨の金額を設定した配列を用意

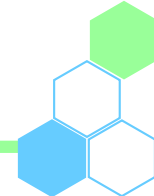
各硬貨の枚数を表す配列

```
int i, coin[] = {500, 100, 50, 10, 5, 1}, coinNum = new int[6];
for (i = 0, i < 6; i = i + 1) {
    coinNum[i] = money / coin[i];
    money = money % coin[i];
}
System.out.println("硬貨の枚数は");
for (i = 0; i < 6; i = i + 1) {
    System.out.println(coin[i] + "円玉: " + coinNum[i] + "枚");
}
System.out.println("です。");
```

500円玉から順に
枚数計算

出力も配列で可能

coinNumは、配列0番に500円玉の枚数、1番に100円玉の枚数...というように枚数が設定された配列になる(for文で書くとすっきり書ける)



課題のフィードバック～フラグ～(1)

◆ ループ文の処理の結果によって、その後の処理内容が変わる場合

◆ 単純にif文では処理ができない

➡ 「フラグ」という考え方を使う
(「フラグを立てる」という言い方をする)

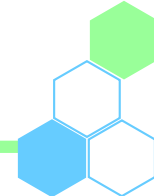
◆ 今回の課題で使えそうなところ

◆ その1で、暗証番号の入力が成功したかどうか

◆ その1で、引き出しの処理が成功したかどうか

◆ その4で、目的の数が見つかったかどうか

◆ etc.



課題のフィードバック～フラグ～(2)

◆ 第1回課題その4

◆ 乱数の中に、入力された数が複数存在するかも

実際のプログラムだと...

➤ 情報検索で、特定のキーワードを含むWebページを探す, etc.

➡ 検索結果が1つだけ、というわけにはいかない

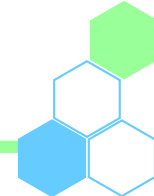
実際に多かったプログラム

```
for(i = 0; i < 100; i = i + 1){  
    if (random[i] == num){  
        i = 101; (または break;)  
    }  
}  
  
if (i < 100) {  
    System.out.println("数" + num + "は" + i + "番目にありました。");  
} else {  
    System.out.println("Not Found");  
}
```

数が見つかったら、ループを強制終了

数が見つかったら、1つだけ出力

課題のフィードバック～フラグ～(3)



◆ フラグの使い方

- ◆ int型の変数を1つ用意する(ここではflagとする)
 - ◆ 変数flagで扱う値が2種類であれば、boolean型でもOK
- ◆ 変数flagに0を代入する
- ◆ ループ文の中で、ある特定の状況になった場合、その状況に応じて1, 2, 3, ...という数をflagに代入する
 - ◆ ループ文の後の条件分岐の条件に応じて、いくつ分の数があれば良いかを定める
- ◆ ループ文の後でif文でflagの値をもとに条件分岐する

ループ文で特定の状況になったときに旗(フラグ)を立てておくことで、ループの終了後に旗がどこかに立っていれば(いなければ)...ということで処理をする



課題のフィードバック～フラグ～(4)

- Ex. 逐次探索で、探したいものが見つからない場合に「Not found!」と出力する

- 探したいものが見つかった、ということで旗を立てる

```
int i, flag = 0;
for (i = 0; i < 100; i = i + 1) {
    if (random[i] == num) {
        System.out.println("数" + num + "は" + i + "番目にありました。");
        flag = 1;
    }
}
if (flag == 0) {
    System.out.println("Not found!");
}
```

数が乱数の中に見つかり、
flagの値を変更

flagの値が0のまま
= 数が見つからなかった

※この場合、「見つからなかった場合」はflagの初期値としない
(for文で、「見つかった」は判断できるが、「見つからなかった」は判断できないため)