

情報処理技法 (Javaプログラミング)1

第8回

同じ処理を何回も繰り返すには？

人間科学科コミュニケーション専攻

白銀 純子

● 第8回の内容

- 同じ処理を何度も繰り返す方法

● 前回の復習問題の解答

- 下記の標準入力をする処理で、間違っている部分を探しなさい。
 - ※間違っているの1ヶ所だけです。import文やクラス宣言、メインメソッドなどは省略しています。

```
String str;  
try {  
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
  
    str = br.readLine();  
}  
System.out.println(str);  
catch (IOException e) {  
    System.out.println("標準入力において例外が発生しました。");  
}
```

この部分が間違い
➤ tryの後、catchの前に文を書いてはならない

● 繰り返し文

● 同じ処理を繰り返す場合 (p. 177)

- 例えば、1～50の番号を表示したい場合



50回分の標準出力が必要!

```
System.out.println("1番");  
System.out.println("2番");  
.....  
System.out.println("50番");  
と書く?
```

面倒!!

しかもやっていることは同じ! (「x番」という形で標準出力しているだけ)

繰り返し(ループ)

繰り返し(ループ)って?(p. 177)

- ある条件を満たしている間、同じ処理を何回もする場合に使う文

- for文
- while文
- do～while文

例えば処理を実行した回数を数え、指定された回数に達しているかどうか、など

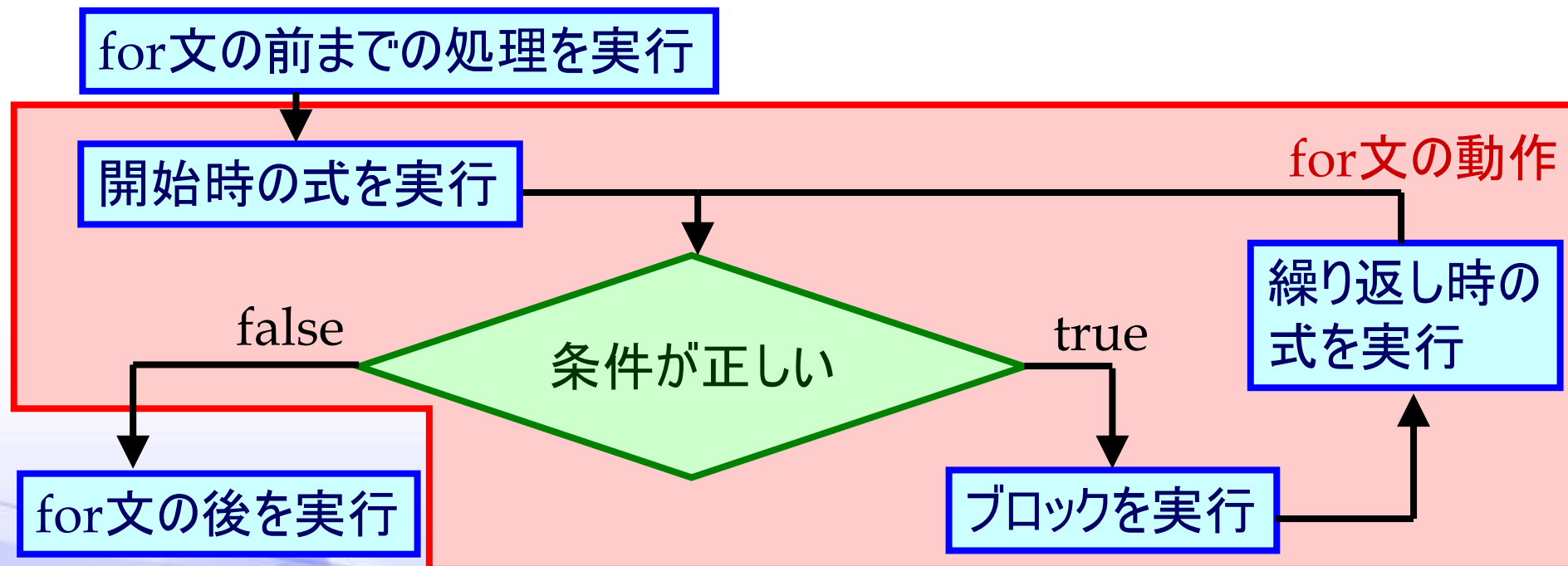
for文(p. 178)

- 主に、処理を繰り返す回数が決まっている場合に利用
 - 配列(この後で)を扱う場合に使うことが多い

for文の書き方と動作(p. 178)

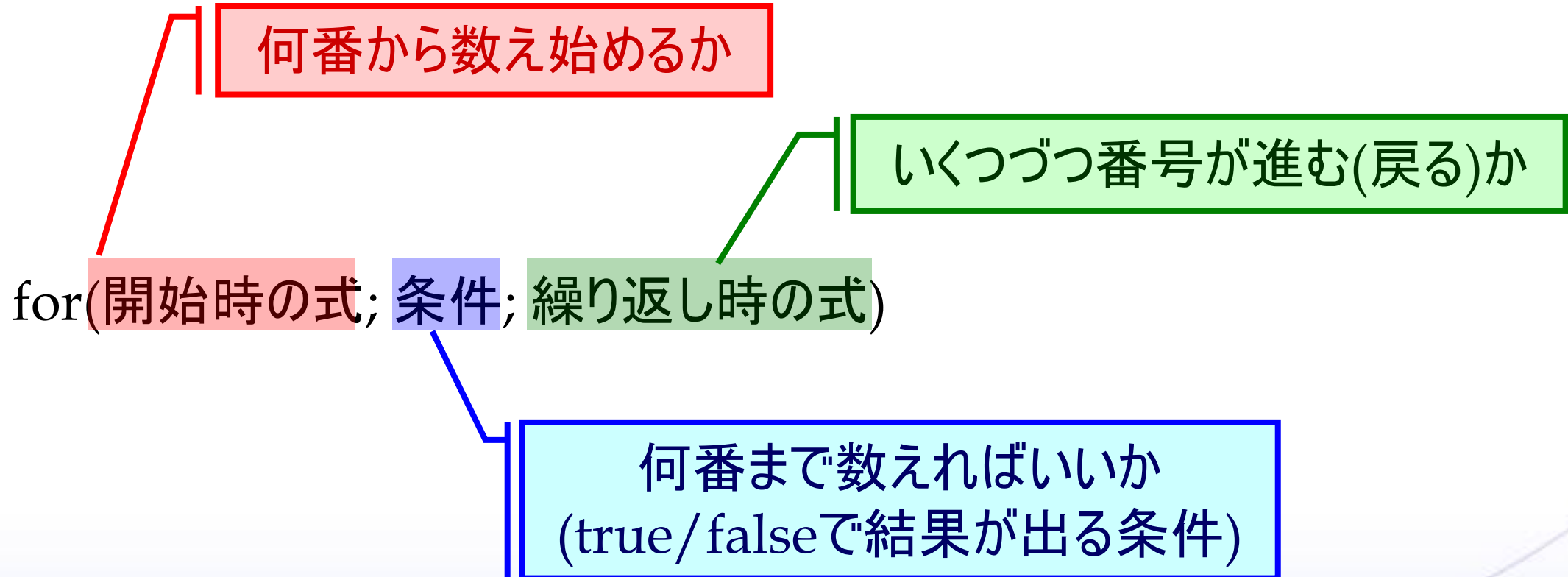
```
for(開始時の式; 条件; 繰り返し時の式) {  
    文1;  
    文2;  
    .....  
}
```

ブロック



for文の一般的な使い方(p. 178)

- 1, 2, 3, ...や、50, 49, 48, ...のように、何かの数を数える場合に利用
 - 繰り返す回数が決まっている場合に利用



※数える数は負数でもOK

for文のよくある使い方(1)

- 多くの数の足し算・引き算・掛け算・割り算

- $1 + 2 + 3 + 4 + 5 + \dots$

- $10 * 10 * 10 * 10 * \dots$

- etc.

- 数の個数が決まっている

- すべての数を同じ方法で計算する

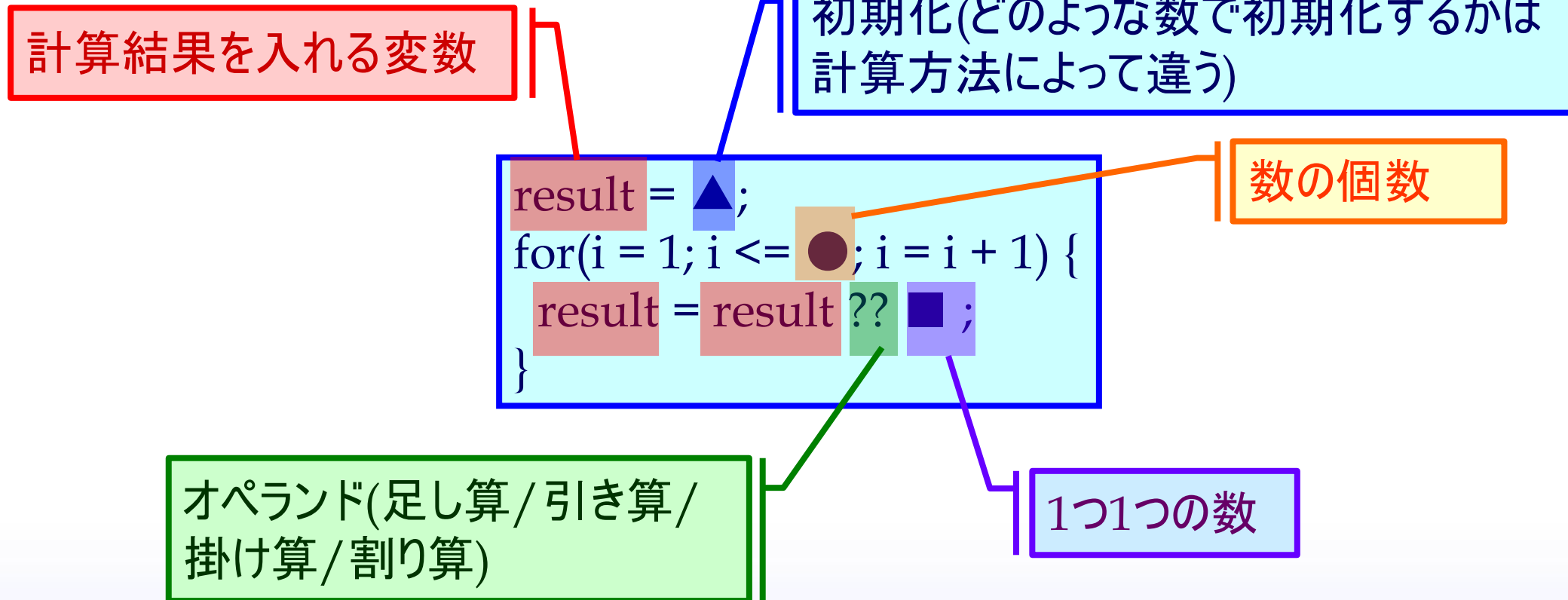
- ✓ Ex. すべての数を足し算する、など

- ✓ 足し算したり掛け算したり、違う方法の計算はしない

➡ for文を使う

for文のよくある使い方(2)

- for分を使った計算のテンプレート



for文の使い方例(1)(p. 179)

- 0から49まで順に数えたい場合

```
for(i = 0; i < 50; i = i + 1)
```

- 0から100までの数のうち、偶数のみを扱う場合

```
for(i = 0; i < 100; i = i + 2)
```

- 49から0まで順に数えたい場合

```
for(i = 49; i > 0; i = i - 1)
```

※「i」はint型の変数

(i, jは、ループ文で数を数えるための変数としてよく使われる)

for文の使い方例(2)(p. 179)

- 1から50までの数を足し合わせた数を求める

```
result = 0;  
for(i = 1; i <= 50; i = i + 1) {  
    result = result + i;  
}
```

「代入の不思議」の復習: 「result=result+i」は、
この式の前までの「result」の値に「i」を足し、
「result」の値を新しくする

for文に入る前: resultの値は0になる
iの値が1の時: resultの値は0 + 1になる
iの値が2の時: resultの値は0 + 1 + 2になる
iの値が3の時: resultの値は0 + 1 + 2 + 3になる
.....
iの値が50の時: resultの値は0 + 1 + 2 + ... + 50になる
iの値が51の時: ループを終わる

for文の使い方例(3)

- 1から50までの数を足し合わせた数を求める

```
result = 0;  
for(i = 1; i <= 50; i = i + 1) {  
    result = result + i;  
}
```

このプログラムでは、たとえばiの値が3のとき...

result = result + i

この「result」の値:
0 + 1 + 2

この「i」の値: 3

➡ resultの値は「0 + 1 + 2 + 3」になる

●while文(p. 183)

- 主に、処理を繰り返す回数が決まっていない場合に利用

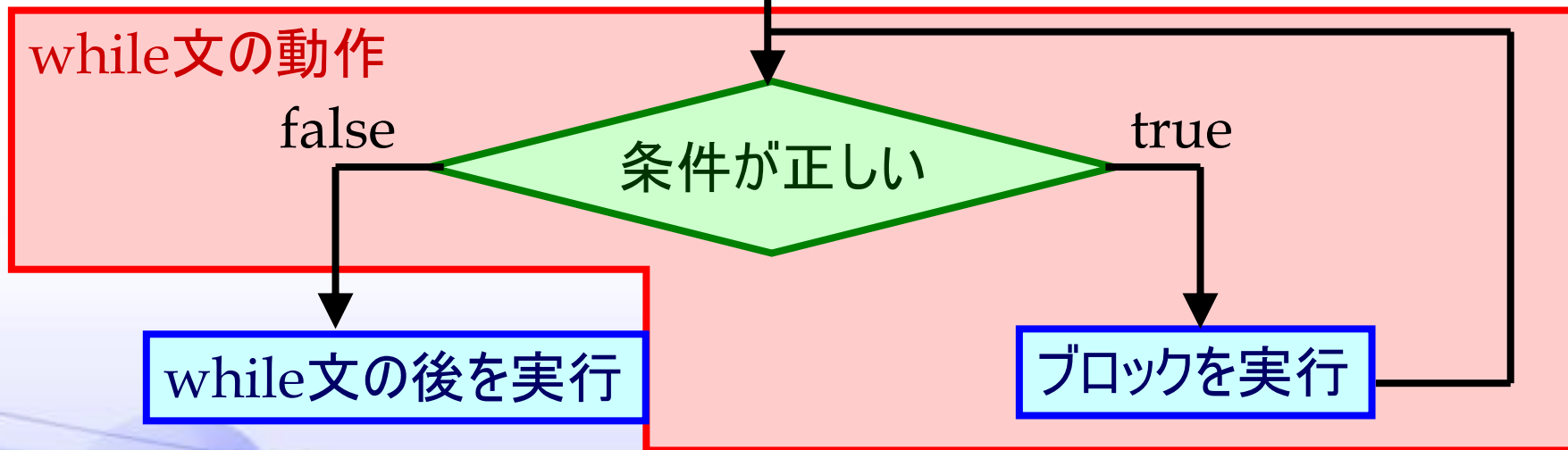
while文の書き方と動作(p. 183)

```
while(条件) {  
    文1;  
    文2;  
    .....  
}
```

ブロック

条件の書き方はif文とまったく同じ
(true/falseで結果が出る条件)

while文の前までの処理を実行



● while文の使い方例(1)(p. 184)

- 1から50までの数を足し合わせた数を求める

```
i = 1;
result = 0;
while (i <= 50) {
    result = result + i;
    i = i + 1;
}
```

while文に入る前: resultの値は0になる
iの値が1の時: resultの値は0 + 1になる
iの値が2の時: resultの値は0 + 1 + 2になる
iの値が3の時: resultの値は0 + 1 + 2 + 3になる
.....
iの値が50の時: resultの値は0 + 1 + 2 + ... + 50になる
iの値が51の時: ループを終わる

※この処理は、for文の例をwhile文に直したもの

● while文の使い方例(2)(p. 184)

- 標準入力から文字列を入力し、その文字列の長さを標準出力で出力する
 - 「End」と入力されれば終了

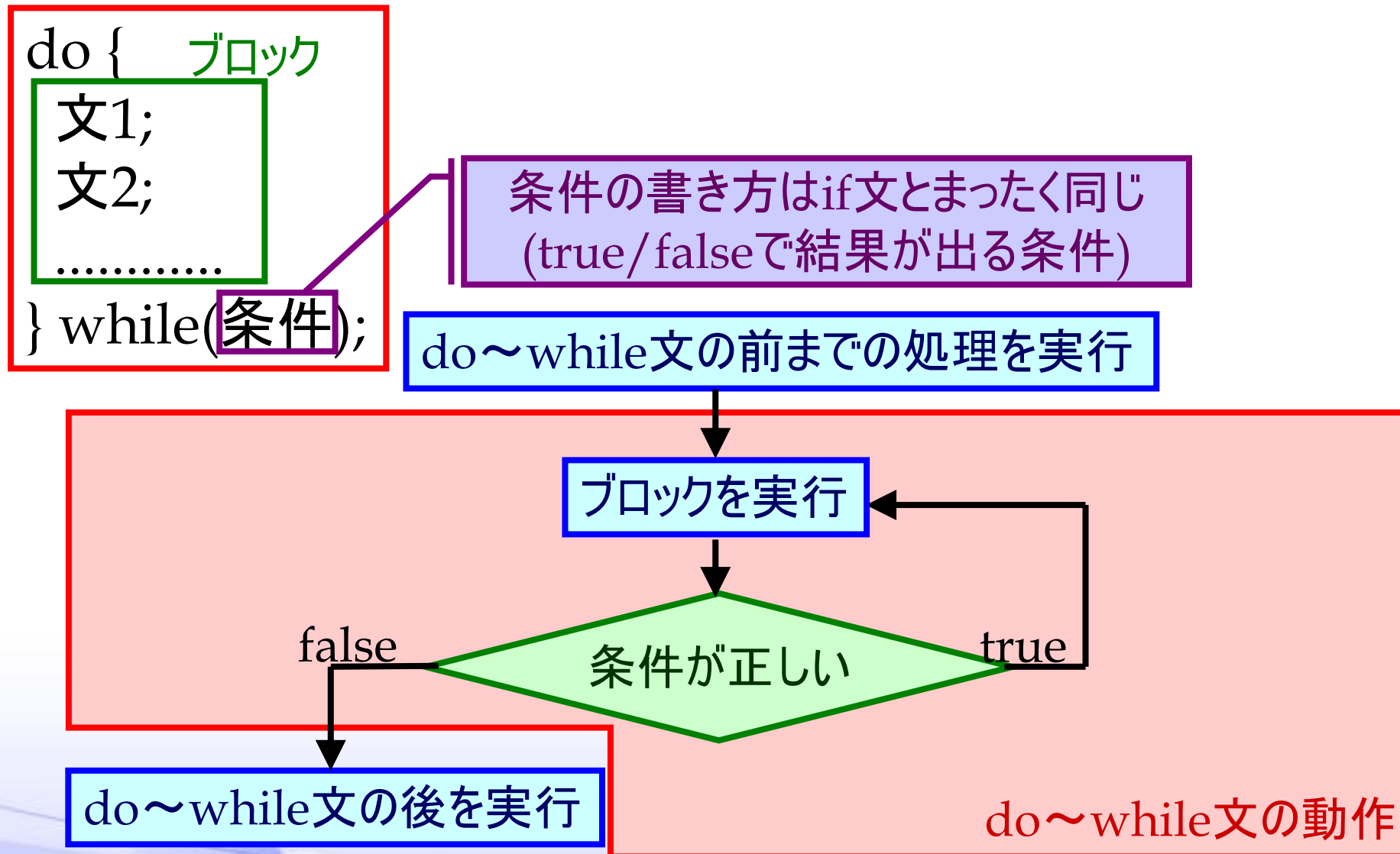
```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
str = br.readLine();
int len;
while (!str.equals("End")) {
    len = str.length();
    System.out.println("入力された文字列の長さ: " + len);
    str = br.readLine();
}
```

※この処理は、for文ではあまり書かない

do～while文(p. 185)

- 主に、繰り返す数が決まっていないときに利用
 - ただし、「～」の部分を1度行ってから条件を判定

do～while文の書き方と動作(p. 185)



do～while文の使い方例(1)(p. 186)

- 1から50までの数を足し合わせた数を求める

```
i = 1;  
result = 0;  
do {  
    result = result + i;  
    i = i + 1;  
} while (i <= 50);
```

do～while文に入る前: resultの値は0になる

iの値が1の時: resultの値は $0 + 1$ になる

iの値が2の時: resultの値は $0 + 1 + 2$ になる

iの値が3の時: resultの値は $0 + 1 + 2 + 3$ になる

.....

iの値が50の時: resultの値は $0 + 1 + 2 + \dots + 50$ になる

iの値が51の時: ループを終わる

do～while文の使い方例(2)(p. 186)

- 標準入力から文字列を入力し、その文字列の長さを標準出力で出力する
 - 「End」と入力されれば終了

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
int len;  
do {  
    str = br.readLine();  
    len = str.length();  
    System.out.println("入力された文字列の長さ: " + len);  
} while (!str.equals("End"));
```

while文とdo～while文の違い

- while文: while文の中の処理は、1度も実行されずに終わることもある

while文の使い方例(2)のプログラム(抜粋)

```
while (!str.equals("End")) {  
    len = str.length();  
    System.out.println("入力された文字列の長さ: " + len);  
    str = br.readLine();  
}
```

標準入力から、最初に「End」と入力されたら、
この部分は実行されない

- do～while文: do～while文の中の処理は、必ず1回は実行される

do～while文の使い方例(2)のプログラム(抜粋)

```
do {  
    str = br.readLine();  
    len = str.length();  
    System.out.println("入力された文字列の長さ: " + len);  
} while (!str.equals("End"));
```

標準入力から、最初に「End」と入力されても、
この部分は1度は実行される

for文とwhileの関係

- for文で書いたものはwhile文でも表現できる
- while文で書いたものはfor文でも表現できる
- ただし、一般的な使い方が限定されている

```
for(i = 0; i < 100; i = i + 1) {  
    result = result + i;  
}
```

```
i = 0;  
while(i < 100) {  
    result = result + i;  
    i = i + 1;  
}
```

同じことを表す

繰り返し文での変数の初期化

- 合計や平均を求める場合には、繰り返しに入る前に、合計を表す変数に「0」を代入しておく
 - 変数は、使い始める前に初期化をしなければならない
 - 「0」を入れておけば、最終的な合計の結果が変わらない
 - 「0」以外の数を入れれば、最終的な合計の結果がおかしくなる
(足し算以外の場合には何が適切か考えること)

```
int result, i  
result = 0;  
for (i = 0; i < 100; i = i + 1) {  
    result = result + i;  
}
```

危険な繰り返し条件(p. 187)

```
for(i = 1; i > 0; i = i + 1) {  
    .....  
}
```

「i」の値は常に正の数だから

```
String str = "abc";  
while (str.equal("abc")) {  
    .....  
}
```

「str」の値は常に「abc」だから

繰り返しが終わらない!!

「無限ループ」と呼ぶ

➡ 無限ループに陥ってしまったら...
「Ctrl + c」(「Ctrl」キーを押しながら「c」キーを押す)

●ループ文の考え方

ループ文の考え方(1)

- 「xx回」のように回数が指定されている場合はループ文が利用可能
 1. 処理内容を3～5回分ほど書き出してみる
 2. 同じことをしている部分を探す
 3. 同じことをしている部分をループ文にまとめる

ループ文の考え方(2)

- Ex. 標準入力で数を100個入力し、その平均を計算するプログラム

```
try {  
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in);  
  
    String str1 = br.readLine();  
    int num1 = Integer.parseInt(str1);  
  
    String str2 = br.readLine();  
    int num2 = Integer.parseInt(str2);  
  
    String str3 = br.readLine();  
    int num3 = Integer.parseInt(str3);  
  
    int sum = num1 + num2 + num3;  
    int average = sum / 3;  
}  
catch(IOException e) {  
}
```

ループ文の考え方(3)

- Ex. 標準入力で数を100個入力し、その平均を計算するプログラム

```
try {  
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in);  
  
    String str1 = br.readLine();  
    int num1 = Integer.parseInt(str1);  
  
    String str2 = br.readLine();  
    int num2 = Integer.parseInt(str2);  
  
    String str3 = br.readLine();  
    int num3 = Integer.parseInt(str3);  
  
    int sum = num1 + num2 + num3;  
    int average = sum / 3;  
}  
catch(IOException e) {  
}
```

} 同じことをしている
= ループ文が利用可能

ループ文の考え方(4)

- Ex. 標準入力で数を100個入力し、その平均を計算するプログラム

```
try {  
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in);  
  
    String str1 = br.readLine();  
    int num1 = Integer.parseInt(str1);  
  
    String str2 = br.readLine();  
    int num2 = Integer.parseInt(str2);  
  
    String str3 = br.readLine();  
    int num3 = Integer.parseInt(str3);  
  
    int sum = num1 + num2 + num3;  
    int average = sum / 3;  
}  
catch(IOException e) {  
}
```

sum1 = num1;

sum1 = num1 + num2;

sum1 = num1 + num2 + num3;

1度の入力ごとに足し算をすると...

ループ文の考え方(5)

- Ex. 標準入力で数を100個入力し、その平均を計算するプログラム

```
int sum;  
String str1 = br.readLine();  
int num1 = Integer.parseInt(str1);  
(1) sum = num1;  
  
String str2 = br.readLine();  
int num2 = Integer.parseInt(str2);  
(2) sum = num1 + num2;  
  
String str3 = br.readLine();  
int num3 = Integer.parseInt(str3);  
(3) sum = num1 + num2 + num3;  
  
int average = sum / 3;
```

- (2)の段階で変数sumに(1)の処理結果が入っている
- (3)の段階で変数sumに(2)の処理結果が入っている



- (2)は「**sum = sum + num2**」と書ける
- (3)は「**sum = sum + num3**」と書ける



「**変数 = 変数 + 入力された数**」の形でループ文にすることが可能

ループ文の考え方(5)

- Ex. 標準入力で数を100個入力し、その平均を計算するプログラム

```
int sum;  
String str1 = br.readLine();  
int num1 = Integer.parseInt(str1);  
sum = num1;  
  
String str2 = br.readLine();  
int num2 = Integer.parseInt(str2);  
sum = num1 + num2;  
  
String str3 = br.readLine();  
int num3 = Integer.parseInt(str3);  
sum = num1 + num2 + num3;  
  
int average = sum / 3;
```

100回繰り返すループ文に
書き直すと...



```
int num, sum, i;  
String str;  
  
for (i = 0; i < 100; i = i + 1) {  
    str = br.readLine();  
    num = Integer.parseInt(str);  
    sum = sum + num;  
}  
int average = sum / 100;
```

●多重ループ(p. 189)

● 多重ループ(p. 189)

- ループ文の中にループ文があるもの

```
for(.....) {  
    for (.....) {  
        for (.....) {  
        }  
    }  
}
```

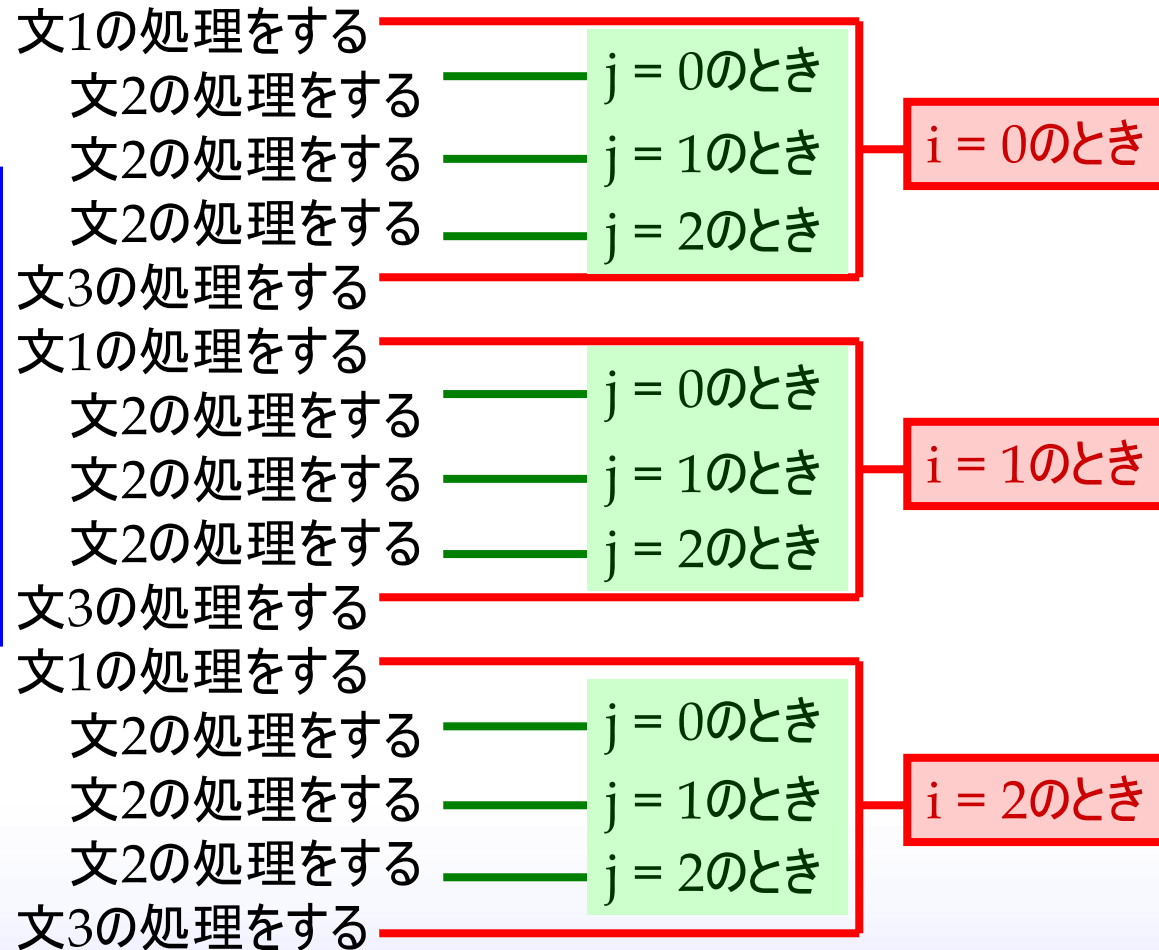
```
while (...) {  
    while (...) {  
        while (...) {  
        }  
    }  
}
```

```
while (...) {  
    for (.....) {  
    }  
}
```

多重ループの動作(p. 189)

例えば...

```
for(i = 0; i < 3; i = i + 1) {  
  文1;  
  for (j = 0; j < 3; j = j + 1) {  
    文2;  
  }  
  文3;  
}
```



※for文でなく、while文、do～while文でも同様に動作

● 多重ループの例(p. 189)

- 数を縦横(縦3行、横4列)に並べて表示するプログラム

プログラム

```
int i, j, num = 1;
for (i = 0; i < 3; i = i + 1) {
    for (j = 1; j <= 4; j = j + 1) {
        System.out.print(num + " ");
        num = num + 1;
    }
    System.out.println("");
}
```

実行結果

```
1 2 3 4
5 6 7 8
9 10 11 12
```

繰り返しを考えるときは...(1)

- 特にfor文の場合

- for文で処理内容を書いてみる
- 何回繰り返しているかを数える変数(iやjなど)が、どういう値のときに処理は具体的にどうなっているかを考えてみる

- 数え始めの数からいくつか分、数え終わりの数からいくつか分くらいを考えてみる

- いくつか例を出して考えると、考えやすくなる
- iやjが数え始めの数や数え終わりの数になったときにエラーが起こりやすい
→ 必ずチェック!!!

for(i = X; i < Y; i = i + 1)

iがXのとき, iがX+1のとき,
iがX+2のとき, ...

iがYのとき, iがY-1のとき,
iがY-2のとき, ...

繰り返しを考えると... (2)

- 1から50までの数を足し合わせた数を求めるfor文

```
result = 0;  
for(i = 1; i <= 50; i = i + 1) {  
    result = result + i;  
}
```

書いたプログラムについて、この程度のことを考えてみる
→うまくいかないところを発見しやすくなる

for文に入る前: resultの値は0になる
iの値が1の時: resultの値は0 + 1になる
iの値が2の時: resultの値は0 + 1 + 2になる
iの値が3の時: resultの値は0 + 1 + 2 + 3になる
.....
iの値が50の時: resultの値は0 + 1 + 2 + ... + 50になる
iの値が51の時: ループを終わる



やってみよう!

- 教科書p. 191の例題01-07をやってみよう