プログラミング基礎Ⅰ
第3回
繰り返し

九州産業大学理工学部

pk@is.kyusan-u.ac.jp

プログラムを書く上での注意

- **一区切りごとに実行**して、エラーがないか確認する！！
  - エラーを放置して先に進むと、どんどんエラーが増えていく・・・
  - 表示が変わるタイミングでは必ず実行して確認すること！！！
- 半角/全角に注意！！
  - 文字列を入力するとき以外は、「日本語入力オフ」で入力する
- エラーが出たら、**エラーの内容を確認**してみる
  - ～～を変数に解決できません：その名前の変数がない
  - 構文エラーがあります。“}” を挿入して…：閉じ括弧が足りない
    - **括弧は開いたらすぐ閉じる！**、対応があることを確認するのが大事

第3回講義概要

- 繰り返し
  - for文
  - if文とfor文の組み合わせ
  - while文
  - for文とwhile文の使い分け

予告：第４回の演習回について

- 演習回は試験に出るような内容は新たに出ないが、演習問題が多く出る
  - 新たな構文は出さないが、実用的な例題を提示
- 自分で考えなければ分からない問題がほとんど
- 事前学習で出来るところまで解いておく
- 良い評価を得たい人は早目に取りくんでおこう

繰り返し同じ処理を行う

- 例えば、 0,1,2,3,4,5,6,7,8,9,10,…, 100
  と表示させるプログラムを考える

```
System.out.println("0,1,2,3,4,5,6,7,8,9,10,11,12,13,…");
```

※省略しているが、100まで繰り返し入力

このような書き方でも実現できるが、非常に大変

- 100程度ならまだ何とかなるが、20000までと言われたら？

変数を利用して少し効率化してみる

```
int i = 0;  //変数 i に最初に表示する値を代入しておく
System.out.print(i + ","); //  iとカンマを連結表示(改行しない)
i ++;  // i を 1増やす

System.out.print(i + ","); //  iとカンマを連結表示(改行しない)
i ++;  // i を 1増やす

System.out.print(i + ","); //  iとカンマを連結表示(改行しない)
i ++;  // i を 1増やす

… これを i が100になるまで繰り返し記述する
```

これはこれで面倒だが、ペーストを繰り返すだけで良くなる

変数を利用して少し効率化してみる

```
int i = 0;  //変数 i に最初に表示する値を代入しておく
System.out.print(i + ","); //  iとカンマを連結表示(改行しない)
i ++;  // i を 1増やす
System.out.print(i + ",");
i ++;  // i を 1増やす
System.out.print(i + ",");
i ++;  // i を 1増やす
… これを i が100になるまで繰り返し記述する
```

少し楽になったが、
同じプログラムを繰り返しているだけ

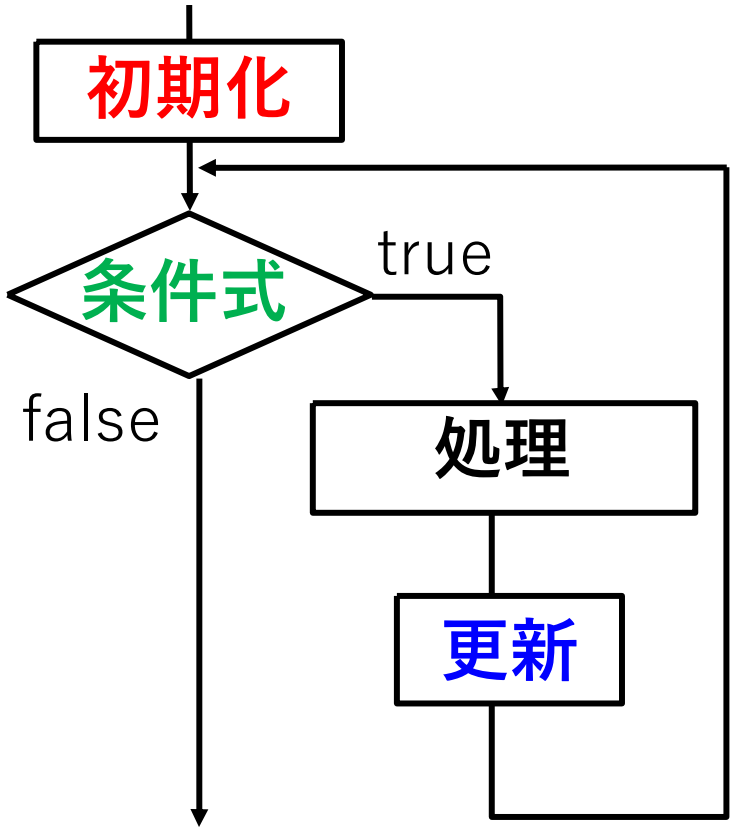
もっと楽にできないのか？

これはこれで面倒だが、ペーストを繰り返すだけで良くなる

for文　教科書P60

- 指定した回数だけ処理を繰り返す
- for文の書き方

```
for(初期化; 条件式; 更新){
        処理
}
```



初期化：ループ変数の初期化（この値から）

条件式：ループを続けるかどうか（この条件が正しい間繰り返す）

更新　：ループ変数の更新（値を増やしたり、減らしたり）

例）0から9までの値を表示

- 0 1 2 3 4 5 6 7 8 9 と値を増やして表示する

初期化
変数宣言と代入
変数iを作り0に

条件式
iが9以下の間
(i<10 でもよい)

更新
iを1ずつ増やす
(i=i+1と同じ)

```
for(int i = 0; i <= 9; i++){
    System.out.print(i + ",");
}
```

変数 i の値

10

繰り返しで増える
変数の値を画面に表示

[実行結果]

0,1,2,3,4,5,6,7,8,9,

0から9まで1ずつ値を増やして繰り返し表示

for文を書き換えてみると…

```
for(int i = 0; i <= 9; i++){
    System.out.print(i + ",");
}
```

for文は、

繰り返し記述する処理を、
変数を利用して簡潔に表記出来る

```
int i = 0;  //初期化
System.out.print(i + ","); //   処理
i ++;  // 更新
System.out.print(i + ","); //   処理
i ++;  // 更新
System.out.print(i + ","); //   処理
i ++;  // 更新
System.out.print(i + ","); //   処理
i ++;  // 更新
System.out.print(i + ","); //   処理
i ++;  // 更新
System.out.print(i + ","); //   処理
i ++;  // 更新
System.out.print(i + ","); //   処理
i ++;  // 更新
System.out.print(i + ","); //   処理
i ++;  // 更新
System.out.print(i + ","); //   処理
i ++;  // 更新
System.out.print(i + ","); //   処理
i ++;  // 更新
```

iが9以下の間繰り返す

for文利用の利点

- 繰り返し何度も行う処理を簡潔に記述できる

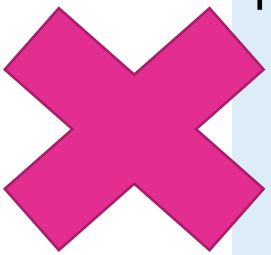
```
for(int i = 0; i < 20000; i++){  //20000回繰り返す
    System.out.print(i + ",");
}
```

- プログラムの**変更が容易**
  - 区切り文字を ，　から ： に変えてくれと言われたら…

```
for(int i = 0; i <= 9; i++){
  System.out.print(i + ":");   //一か所変更するだけで良い
}
```

変数の有効範囲（変数のスコープ）

- 変数はどこでも使えるわけではない
- プログラムのブロック単位で区切られている
  - 同じブロックの中では、同じ名前の変数は宣言できない



```
public static void main(String[] args){
    int x = 0;
    int x = 10;
}
```

同じ名前の変数を宣言しようとしている

これはエラーになる

変数の有効範囲（if文の場合）

- if文のブロック内で宣言された変数は、
  そのif文のブロック内でのみ有効

```
public static void main(String[] args){
    if ( 10 == 10) {
        int x = 0;  //この x はif文の中でのみ有効
    } // if 文が終わると、 ↑の x は使えなくなる(以降は使えない)
    int x = 10;  //この x は if文のブロックのx とは違う新しい x
}
```

これはエラーにはならない

変数の有効範囲（if文の場合）

- if文のブロック内で宣言された変数は、
  そのif文のブロック内でのみ有効

```
public static void main(String[] args){
    int x = 10;  // 先に上のブロックで宣言していると
    if ( 10 == 10) {
        int x = 0;  //この x は宣言できない
    }
}
```

エラーになる

変数の有効範囲（for文の場合）

- for文の中で宣言された変数も、for文のブロック内でのみ有効

```
for(int i = 0; i < 5; i++){
    System.out.println(”Hello”);
}
for(int i = 0; i < 5; i++){
    System.out.println(”Hello”);
}
```

for文のループ用変数は、
使い捨てていくイメージ

同じ名前のループ変数を何度も使える

同じ変数を使っているようにみえるが、
それぞれの i はそれぞれ別ものなのでエラーにはならない

例）画面に5回Helloと表示する

- for文でiの値を0 1 2 3 4 と増やしながら、Helloを表示する

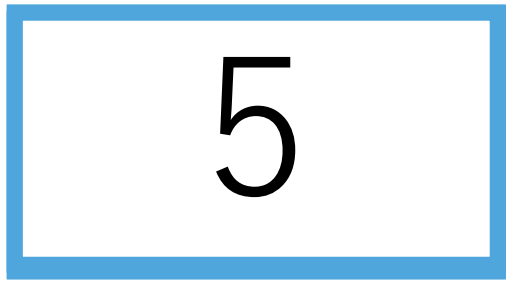
初期化
変数宣言と代入
変数iを作り0に

条件式
iが5より小さい間
（i<=4 でもよい）

更新
iを1ずつ増やす
（i=i+1と同じ）

```
for(int i = 0; i < 5; i++){
    System.out.println(”Hello”);
}
```

変数 i の値



[実行結果]

Hello
Hello
Hello
Hello
Hello

0から4まで1ずつ値を増やして繰り返し表示

例）0から9までの値を表示

- 0 1 2 3 4 5 6 7 8 9 と値を増やして表示する

初期化
変数宣言と代入
変数iを作り0に

条件式
iが9以下の間
(i<10 でもよい)

更新
iを1ずつ増やす
(i=i+1と同じ)

```
for(int i = 0; i <= 9; i++){
    System.out.print(i);
}
```

変数 i の値

10

繰り返しで増える
変数の値を画面に表示

[実行結果]

0123456789

0から9まで1ずつ値を増やして繰り返し表示

for文を使ったプログラムの記述例

```
public class Test {
  public static void main(String[] args)
    for( int i = 0; i < 10; i++){
      System.out.print(“#”);
    }
    System.out.println( );
    for( int i = 3; i < 10; i += 2){
      System.out.print( i + “,“);
    }
  }
}
```

System.out.print**ln**()は
表示して改行
System.out.print()は
改行せずに表示

横に並べて表示したい時は
System.out.print()で
値を表示し、繰り返し終了後に
System.out.print**ln**()で改行

[実行結果]
##########
3,5,7,9,

同じiの名前のループ変数を繰り返し使っても良い

for文

```
for(  int i =  0 ; i < 6  ; i++   ) {
   System.out.print (   "*"   );
}
```

[実行結果]
******

```
for(  int j =  3 ; j <= 18; j+=3  ) {
   System.out.print (   j+", "   );
}
```

[実行結果]
3, 6, 9, 12, 15, 18,

```
for(  int t = 30 ; t >= 5 ; t-=5   ) {
   System.out.print (  "#" +t     );
}
```

[実行結果]
#30#25#20#15#10#5

※改行して表示したい場合には、
各for文の後にSystem.out.println();
が必要

穴埋め：考えてみよう

```
for( int i =  0 ; i < 5  ; i++ ) {
   System.out.print ( i + "," );
}
System.out.println();
```

このプログラムを実行したときに表示される結果として正しいものはどれか？

正しいものに印を付けよ

__ 1,2,3,4,5

__ 0,1,2,3,4,5,

__ 0,1,2,3,4,

__ 0,1,2,3,4

__ 1,5,10,15,20,

穴埋め：考えてみよう

```
for( int i =  0 ; i < 5  ; i++ ) {
   System.out.print ( i + "," );
}
System.out.println();
```

このプログラムを実行したときに表示される結果として正しいものはどれか？

正しいものに印を付けよ

__ 1,2,3,4,5

__ 0,1,2,3,4,5,

○ 0,1,2,3,4,

__ 0,1,2,3,4

__ 1,5,10,15,20,

穴埋め：考えてみよう

```
for(  int i =  20 ; i >= -5  ; i-=5 ) {
   System.out.print (   i + ","   );
}
System.out.println();
```

このプログラムを実行したときに表示される結果として正しいものはどれか？

正しいものに印を付けよ

__ 20,15,10,5,0

__ 0, 5,10,15,20,

__ 20,15,10,5,0,-5

__ 20,15,10,5,0,-5,

__ 1,5,10,15,20,

穴埋め：考えてみよう

```
for(  int i =  20 ; i >= -5  ; i-=5 ) {
   System.out.print (   i + ","   );
}
System.out.println();
```

このプログラムを実行したときに表示される結果として正しいものはどれか？

正しいものに印を付けよ

__ 20,15,10,5,0

__ 0, 5,10,15,20,

__ 20,15,10,5,0,-5

○ 20,15,10,5,0,-5,

__ 1,5,10,15,20,

for文の例

```
for(  int i =  0 ; i < 5  ; i++ ) {
   System.out.print (   "*"   );
}
```

*を5個横に表示
[実行結果]
******

```
for( int i = 1; i <= 6 ; i += 1   ) {
   System.out.print ( i + "#" );
}
```

1から6まで1区切り
[実行結果]
1#2#3#4#5#6#

```
for( int a = 1; i <= 15 ; a += 2) {
   System.out.print ( a + "," );
}
```

1から15まで2区切り
[実行結果]
1,3,5,7,9,11,13,15,

※改行して表示したい場合には、
各for文の後にSystem.out.println();
が必要

穴埋め：for文の例

```
for( int t = 30; t >= -10 ; t-= 5   ) {
    System.out.print ( t + " " );
}
System.out.println();
```

30から-10まで5区切り
[実行結果]
30 25 20 15 10 5 0 -5 -10

String型変数に繰り返しで文字列を連結

- 前ページのfor文は

```
String dispStr = "";  // 空文字を代入
for( int t = 30; t >= -10 ; t-= 5   ) {
    dispStr += t + " ";
}
System.out.println(dispStr);
```

こう書き換えることも出来る.

System.out.printで順々に表示すると、ループ変数の変化がわかりやすいが、こちらの書き方のほうが応用的に扱える

if文とfor文：1から10までの偶数奇数の判定

```
public class For2 {
    public static void main(String[] args) {
        for(int i = 1; i <= 10; i++) { //1から10
            System.out.print(i+" ");
            if (i % 2 == 0) {    //もし偶数なら
                System.out.println("偶数");
            } else {             //それ以外なら
                System.out.println("奇数");
            }
        }
    }
}
```

{ }の対応に注意！！

処理のかたまりを意識しよう

[実行結果]
1 奇数
2 偶数
3 奇数
…
10 偶数

例）if文とfor文：#-#-#-#-#　と表示

偶数は#で奇数が-であるのに注目

```
for(int i = 0; i < 10; i++) {  // 0 ~ 9 まで繰り返し
    if (i % 2 == 0) {  //もしも、iが偶数なら
        System.out.print(“#”);   //# を表示する
    } else {                    //それ以外なら(奇数なら)
        System.out.print(“-”);  // - を表示する
    }
}
System.out.println();
```

インデントを付ける意味

- 各構文のブロックを分かりやすくするため
  - for文のブロックは？　見やすいのはどっち？

```
for(int i = 0; i < 10; i++)
  if (i % 2 == 0) {  //もしも、iが偶数なら
    System.out.print(“#”);   //# を表示する
  } else {
    System.out.print(“-”);  // - を表示する
  }
}
System.out.println();
```

```
for(int i = 0; i < 10; i++)
if (i % 2 == 0) {  //もしも、iが偶数なら
  System.out.print(“#”);   //# を表示する
} else {
System.out.print(“-”);  // - を表示する
}
}
System.out.println();
```

- 可読性を保つために、
  正しいインデントでプログラミングすることは**必須**

穴埋め：偶数奇数判定

- if文とfor文を組み合わせて、次の表示をしなさい
  - **＊ 2 ＊ 4 ＊ 6 ＊ 8 ＊ 10**

```
for(  int i = __; ____  ; ____ ) {  //1～10まで繰り返す
   if ( _____________ ) {  // もしi が偶数なら
       System.out.print( i );  //変数の値をそのまま表示
   _____________  //そうでないなら(iが奇数なら)
     System.out.print ( "*" );  //*を表示
  }
}
```

穴埋め：偶数奇数判定

- if文とfor文を組み合わせて、次の表示をしなさい
  - **＊ 2 ＊ 4 ＊ 6 ＊ 8 ＊ 10**

```
for(  int i = 1; i <= 10  ; i++ ) {  //1～10まで繰り返す
   if ( i % 2 == 0 ) {  // もしi が偶数なら
       System.out.print( i );  //変数の値をそのまま表示
   } else {  //そうでないなら(iが奇数なら)
     System.out.print ( "*" );  //*を表示
  }
}
```

穴埋め：剰余を利用した表示

- if文とfor文を組み合わせて、次の表示をしなさい
  - **1*#4*#7*#**

```
for( int i = 1; ____  ; i++ ) {  //1～? まで繰り返す
   if ( _____________ ) {  // iを3で割った余りが1なら
    System.out.print( i );  //変数の値をそのまま表示
  _____________________  // iを3で割った余りが2なら
    System.out.print ( "*" );  //*を表示
  _____________________  //そうでないなら
    System.out.print ( "#" );  //#を表示
}
```

穴埋め：剰余を利用した表示

- if文とfor文を組み合わせて、次の表示をしなさい
  - **1*#4*#7*#**

```
for( int i = 1; i <= 9  ; i++ ) {  //1～9 まで繰り返す
   if ( i % 3 == 1 ) {  // iを3で割った余りが1なら
    System.out.print( i );  //変数の値をそのまま表示
  } else if(i % 3 == 2) {  // iを3で割った余りが2なら
    System.out.print ( "*" );  //*を表示
  } else {  //そうでないなら
    System.out.print ( "#" );  //#を表示
}
```

二重ループ 教科書P64

- for文も「文」の一種なので、for文の「文」の中に別のfor文を書くことができる

```
for(int i = 1; i <= 3; i++){
  for(int j = 1; j <= 3; j++){
    System.out.print(i+”-”+j+” “);
  }
 System.out.println();
}
```

[実行結果]
1-1 1-2 1-3
2-1 2-2 2-3
3-1 3-2 3-3

穴埋め：２重ループ

- 実行結果のようになるように、２重ループのfor文を完成さ

```
for(int i = 0; _____; i++) {  // 縦に4つ並べたい・・・
        for(int j = 0; _____; j++) {  // 横に3つ並べたい・・・
                System.out.print("*");
        }
        System.out.println();
}
```

[実行結果]
***
***
***
***

穴埋め：２重ループ

- 実行結果のようになるように、２重ループのfor文を完成さ

```
for(int i = 0; i < 4; i++) {  // 縦に4つ並べたい・・・
        for(int j = 0; j < 3; j++) {  // 横に3つ並べたい・・・
                System.out.print("*");
        }
        System.out.println();
}
```

[実行結果]
***
***
***
***

穴埋め：２重ループで階段状に表示する

- 実行結果のようになるように、２重ループのfor文を完成さ

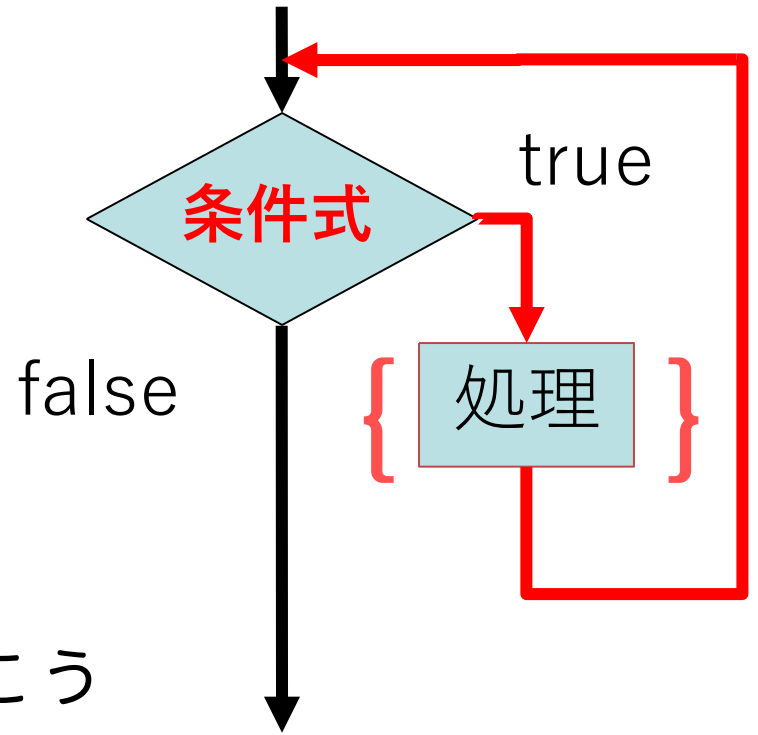
```
for(int i = 1; _____; i++) {  // 縦に4つ並べたい・・・
        for(int j = 1; _____; j++) {  // 横に？個並べたい・・・
                System.out.print("*");
        }
        System.out.println();
}
```

[実行結果]
*
**
***
****

while文　教科書P66

- ある条件が満たされている間処理を繰り返す
- while文の書き方

```
while(条件式){
        処理
}
```



例：雨が降っている間(while)は家で本でも読んでおこう

```
while ( tenki == AME ) {
  book();
}
```

while文とfor文の使い分け

- 実行する回数が決まっているプログラムはfor文、
  そうでないプログラムではwhile文を使う

- while文を使う例
  - ユーザ入力で終了させるようなプログラム
  - ファイルを読み込むプログラム
  - そもそも、終了しないプログラム（無限ループ）
    - ずっと動き続けていて欲しい場合など
  - 繰り返し中に、繰り返し回数が変わるプログラム

繰り返しの中断(終了)

- break;
  - break の命令が実行されると、
    for, while,switch文のブロックを抜けて繰り返しを終了させる
  - 繰り返しを終了させる時に用いる
- continue;
  - continue の命令が実行されると、
    繰り返し中の処理をそこで終了し、
    次の繰り返し処理に移る
  - その繰り返しの処理をスキップしたい時に用いる

ユーザ入力で繰り返しを終了

- 繰り返し中にキーボード入力を受付、
  条件を満たしている間、繰り返す

```
int answer = 1;
while ( answer == 1 ) { // answerが1の間繰り返す
    System.out.print("Hello, world. more ?");
    answer = sc.nextInt();
 }
System.out.println("End");
```

無限ループを中断するプログラム

- 事前に繰り返し回数がわからない場合などは、
  繰り返しの途中で、条件を満たした場合にbreakして終了

```
System.out.println("会計開始");
int totalPrice = 0; // 会計の合計を格納する変数を宣言
while ( true ) { // 無限ループ
        System.out.print("価格を入力: ");
        int price = sc.nextInt();
        if(price <= 0) { // 0以下が入力されたら
                System.out.println("合計を終了");
                break; // ここで繰り返しを抜ける
        }
        totalPrice += price; // 合計価格に加算
}
System.out.println("会計は" + totalPrice + "円");
```

```
会計開始
価格を入力: 100
価格を入力: 200
価格を入力: 300
価格を入力: -1
合計を終了
会計は600円
```

無限ループ(停止しない繰り返し) 教科書P67

- 故意に無限ループを作りたい場合

```
while ( true ) { ...
    trueの間つまり常に繰り返す
}
```

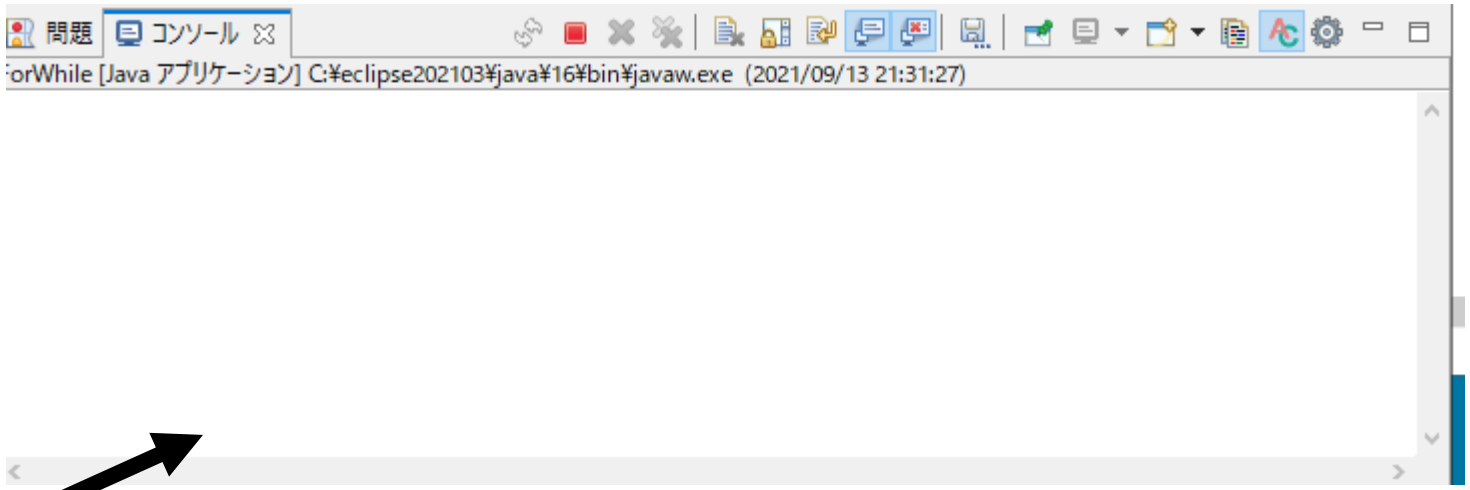
（止めたくないプログラムはあえて無限ループにする。）

ifとbreak(教科書P70)を組み合わせて繰り返しから抜け出すことも可能

意図せずにプログラムが繰り返しを続けた時に止める方法

```
for(int i = 1; i < 10; i--) { // i++を間違えてi--に
    System.out.println(i);
}
```

間違ったfor文を書いて、
繰り返しがいつまでたっても終わらない・・・



実行結果

```
1
0
-1
-2
... 強制終了したい
```

Eclipseの場合は、赤い四角形をクリック

オンライン小テストでのfor文記述のルール

- if文と同じく、条件式は主体となるものを左辺に書く
- 3,6,9,12, と表示、のような問題の場合には、
  例） `for(int i = 3; i <= 12; i+=3)` は正解
  `for(int i = 3; i < 13; i+=3)` などは不正解
  - 開始と終了の値をfor文中に記述する書き方のみ正解
- * を5個並べて表示、のような問題の場合には、

```
for(int i = 0; i < 5; i++)  は正解
```

```
for(int i = 1; i <= 5; i++)  は不正解
```

正解としている場合もあるが、全てではない

まとめ

- for文：決まった回数処理を繰り返す

```
for(初期値; 条件; 更新){
    処理
}
```

- while文：条件を満たしている間繰り返す

```
while(条件) {
    処理
}
```