プログラミング入門1

第5回

繰り返し(whileループ)

授業開始前に

ログオン後、

不要なファイルを削除して待機してください

参考書について

- 参考書は自分にあったものをぜひ手元において自習してください。
  - 授業のWEB教材は勉強の入り口へみなさんを案内するのが目的でつくられている。これで十分という訳ではない。
  - 第１回に紹介した本以外にも良書がたくさんある。

ソースプログラムの健康について

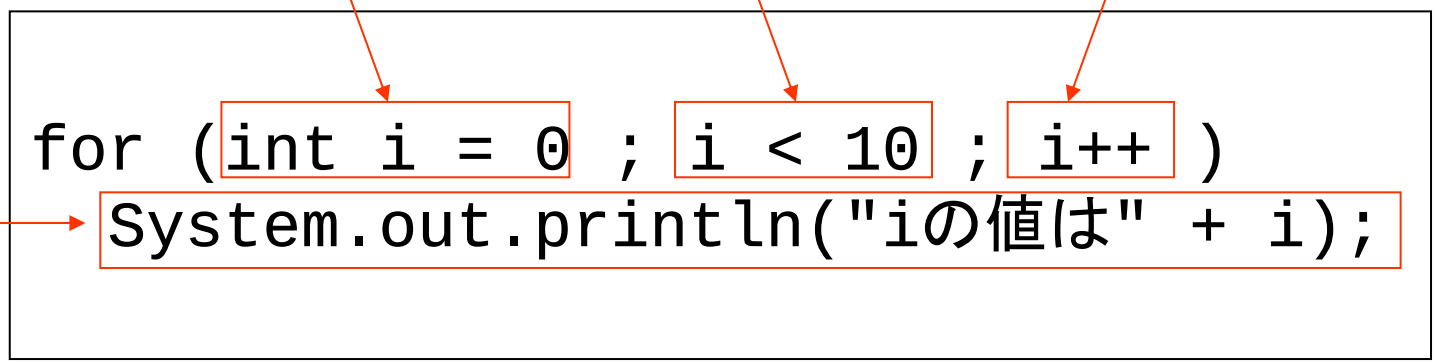
- ソースプログラムは常に健康な状態に保ってください。
  - 健康な状態とは、
  - インデント（字下げ）が適切にされていて、
  - 論理が一目で分かる状態をいう。
- Eclipseでは清書の機能があるが文法誤りがあるとだめ
  - 命令文、ブロック単位でこまめに保存し、誤りが蓄積しないようにするとよい

前回のテーマ

- 繰り返し実行のための便利な書き方
- for 文を用いる

for 文の構造

```
for (<初期化式> ; <条件式> ; <ステップを進める式>)
   文
```



ループ継続の判定

- 毎回最初に条件式がtrueかfalseかチェックされる
- trueであればその回を実行する
- falseであれば、その回は実行されず、for文の繰り返し実行自体も打ち切られる

```
for (int i = 0 ; i < 10 ; i++ )
  System.out.println("iの値は" + i);
```

ステップの進め方いろいろ

```
for (int i = 0 ; i < 10 ; i++ )
  System.out.println("iの値は" + i);
```

```
iの値は0
iの値は1
iの値は2
iの値は3
iの値は4
iの値は5
iの値は6
iの値は7
iの値は8
iの値は9
```

```
for (int i = 0 ; i < 10 ; i+=2 )
  System.out.println("iの値は" + i);
```

```
iの値は0
iの値は2
iの値は4
iの値は6
iの値は8
```

カウントダウンも活用しよう

```
for (int i = 0 ; i < 10 ; i++ )
  System.out.println("iの値は" + i);
```

```
iの値は0
iの値は1
iの値は2
iの値は3
iの値は4
iの値は5
iの値は6
iの値は7
iの値は8
iの値は9
```

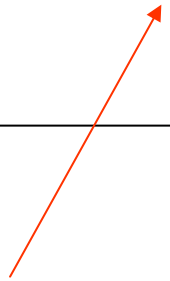
```
for (int i = 9 ; i >=0 ; i-- )
  System.out.println("iの値は" + i);
```

```
iの値は9
iの値は8
iの値は7
iの値は6
iの値は5
iの値は4
iの値は3
iの値は2
iの値は1
iの値は0
```

ユーザが指定した回数だけ繰り返す

```
BufferedReader reader
  = new BufferedReader(new InputStreamReader(System.in));
int input = Integer.parseInt(reader.readLine());

for (int i = 0; i < input ; i++) {
 (命令文の列)
}
```



変数がきてもよい

ループ制御の変数iと役割が違う

今回のテーマ

- while 文を用いた繰り返し実行
  - for文との使い分け
- 複雑な条件判定
  - 「かつ」「または」 を使って

while 文の意味と構造

条件式がtrueの間繰り返す

```
while (条件式)
   文
```

繰り返し実行したい文が複数のときはブロックにする

```
while (条件式) {
   文の列
}
```

while文はfor文から「初期化式」を外に出し、「ステップを進める式」を 繰り返し実行される文に移したもの

```
for (int i = 1; i < 10; i++) {
   System.out.println("i = " + i);
}
```

```
int i = 1;
while ( i < 10 ){
   System.out.println("i = " + i);
   i++;
}
```

for文、while文それぞれの使いどころ

ループに入る時点で繰り返しの回数が決まっているときは

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

繰り返しの回数が指定できないとき

```
...
int input;
input = Integer.parseInt(reader.readLine());
while (input != 0) {
  System.out.println(input);
  input = Integer.parseInt(reader.readLine());
}
```

ステップを進める式（ループ制御変数の値を更新する式）はいろいろな姿をとる

番兵(sentinel)によるループの制御

```
...
int input;
input = Integer.parseInt(reader.readLine());
while (input != 0) {
   System.out.println(input);
   input = Integer.parseInt(reader.readLine());
}
```

番兵inputが0でない間繰り返す

ループに入る前の入力で0だとループは
1度も実行されない

for文に相応しい問題

10000円を利率5%の複利で預金した際の、10年後の預金額を計算する

```
// 元本
double amount = 10000;
// 10回繰り返す (0年後から始めて; 10年後まで; 1年ずつ考える)
for (int year = 0; year < 10; year++) {
  amount *= 1.05;
}
System.out.println("10年後は" + amount + "円");
```

実行結果

```
10年後は16288.946267774418円
```

while文に相応しい問題

10000円を利率5%の複利で預金した際に、預金額が20000円を超えるまでの年数を計算する

```
double amount = 10000; // 元本
int year = 0; // 開始した時点では0年後

// 20000円を超えるまで繰り返す (= 20000円以下のうちは繰り返す)
// → 20000円以下であることを監視する番兵を置く

while (amount <= 20000) {
  // 1年後に利子をつける
  year++;
  amount *= 1.05;
}

System.out.println(year + "年後に20000円を超える");
```

実行結果

```
15年後に20000円を超える
```

do-while文

- というwhile文に似たものがあってプログラムが短く書けることがある。
- ここでは説明しない。WEB教材の説明を読んでおくこと。

AND演算子

javaではこのよう
には書けない

```
0 <= a < 10
```

このようなときは、次のように分割して考える。

```
0 <= a && a < 10
```

実際、こんなふうに使う

```
if (0 <= a && a < 10) {
  System.out.println("aの値が0以上10未満");
}
```

3つ以上つなげてもよい

```
0 <= a && a < 10 && a != 5
```

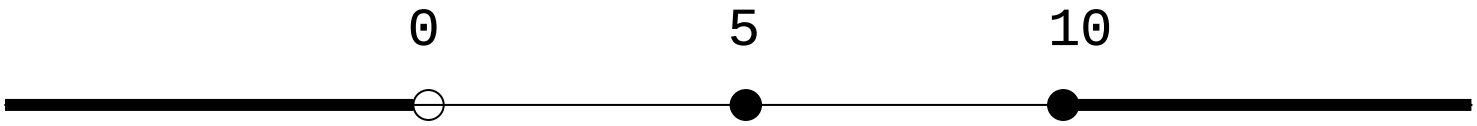
OR演算子 (実際はor-else)

"a<0 または 10 <= a" のような条件は || で結ぶ


```
if (a < 0 || 10 <= a) {
  System.out.println("aの値が0未満 または 10以上");
}
```

3つ以上続けてもよい

```
a < 0 || 10 <= a || a == 5
```



複雑な条件

このような範囲に入るという条件は

```
(0 <= a && a < 10) || (20 <= a && a < 30)
```

&&の方が||より優先順位が高いので以下の式と同等

```
0 <= a && a < 10 || 20 <= a && a < 30
```

NOT演算子


```
0 <= a && a < 10
```

の否定は!


```
!(0 <= a && a < 10)
```

これは次の式と同等

```
a < 0 || 10 <= a
```

演算子の優先順位の注意

```
!a < 0
```

と書くと

```
(!a) < 0
```

と解釈されてコンパイルエラー

```
!(a < 0)
```

これならOK

参考:演算子の優先順位表

高い

優先順位	項数	結合方向	演算子	意味
13	単項	なし	++	変数の中身を1だけ増やす
13	単項	なし	--	変数の中身を1だけ減らす
13	単項	右方向から	!	論理否定(true->false, false->true)
12	二項	左から	*	乗算
12	二項	左から	/	除算
12	二項	左から	%	剰余
11	二項	左から	+	加算
11	二項	左から	-	減算
9	二項	なし	>	より大きい
9	二項	なし	<	より小さい
9	二項	なし	>=	以上
9	二項	なし	<=	以下
8	二項	なし	==	等しい
8	二項	なし	!=	等しくない
4	二項	左から	&&	かつ(and also)
3	二項	左から	||	または(or else)
1	二項	右から	=	代入

一緒にやってみよう

- 今回の演習で使うテストドライバをいつものようにインストールする
  - テストドライバの導入に成功すると
    - プロジェクト「java20XX」の中の「test」というフォルダに「j1.lesson05.xml」という名前のファイルが作成される。
    - このファイルには今週使用するテスト一式が記述されている。
- j1.lesson05 というパッケージを作成する
- WEB教材にあるComplexCondition, UntilLoopというプログラムを、このパッケージに作成する
  - WEB教材にある手順でテスト、実行までやること

ComplexConditionの解説

```
public class ComplexCondition {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    System.out.print("整数を入力:");
    int a = Integer.parseInt(reader.readLine());

    if (0 <= a && a < 10)
      System.out.println("0以上かつ10未満");
    else if (a < -10 || a >= 20)
      System.out.println("-10未満または20以上");

    if (0 < a && ((a % 5) == 0 || (a % 7) == 0))
      System.out.println("5の倍数または7の倍数");
  }
}
```

変数aに整数を入力する

```
public class ComplexCondition {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    System.out.print("整数を入力:");
    int a = Integer.parseInt(reader.readLine());

    if (0 <= a && a < 10)
      System.out.println("0以上かつ10未満");
    else if (a < -10 || a >= 20)
      System.out.println("-10未満または20以上");

    if (0 < a && ((a % 5) == 0 || (a % 7) == 0))
      System.out.println("5の倍数または7の倍数");
  }
}
```


```
public class ComplexCondition {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    System.out.print("整数を入力:");
    int a = Integer.parseInt(reader.readLine());

    if (0 <= a && a < 10)
      System.out.println("0以上かつ10未満");
    else if (a < -10 || a >= 20)
      System.out.println("-10未満または20以上");

    if (0 < a && ((a % 5) == 0 || (a % 7) == 0))
      System.out.println("5の倍数または7の倍数");
  }
}
```


```
public class ComplexCondition {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    System.out.print("整数を入力:");
    int a = Integer.parseInt(reader.readLine());

    if (0 <= a && a < 10)
      System.out.println("0以上かつ10未満");
    else if (a < -10 || a >= 20)
      System.out.println("-10未満または20以上");

    if (0 < a && ((a % 5) == 0 || (a % 7) == 0))
      System.out.println("5の倍数または7の倍数");
  }
}
```

a は正の数の場合

```
public class ComplexCondition {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    System.out.print("整数を入力:");
    int a = Integer.parseInt(reader.readLine());

    if (0 <= a && a < 10)
      System.out.println("0以上かつ10未満");
    else if (a < -10 || a >= 20)
      System.out.println("-10未満または20以上");

    if (0 < a && ((a % 5) == 0 || (a % 7) == 0))
      System.out.println("5の倍数または7の倍数");
  }
}
```

a を5で割った余りが0
つまり、a が5の倍数

```
public class ComplexCondition {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    System.out.print("整数を入力:");
    int a = Integer.parseInt(reader.readLine());

    if (0 <= a && a < 10)
      System.out.println("0以上かつ10未満");
    else if (a < -10 || a >= 20)
      System.out.println("-10未満または20以上");

    if (0 < a && ((a % 5) == 0 || (a % 7) == 0))
      System.out.println("5の倍数または7の倍数");
  }
}
```

a を7で割った余りが0
つまり、a が7の倍数

```
public class ComplexCondition {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    System.out.print("整数を入力:");
    int a = Integer.parseInt(reader.readLine());

    if (0 <= a && a < 10)
      System.out.println("0以上かつ10未満");
    else if (a < -10 || a >= 20)
      System.out.println("-10未満または20以上");

    if (0 < a && ((a % 5) == 0 || (a % 7) == 0))
      System.out.println("5の倍数または7の倍数");
  }
}
```

a が5の倍数である
または a が7の倍数である

```
public class ComplexCondition {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    System.out.print("整数を入力:");
    int a = Integer.parseInt(reader.readLine());

    if (0 <= a && a < 10)
      System.out.println("0以上かつ10未満");
    else if (a < -10 || a >= 20)
      System.out.println("-10未満または20以上");

    if (0 < a && ((a % 5) == 0 || (a % 7) == 0))
      System.out.println("5の倍数または7の倍数");
  }
}
```

UntilLoopの解説

```
public class UntilLoop {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    // 入力を保存する変数を宣言
    int input;
    // 最初の入力
    System.out.print("0で終了:");
    input = Integer.parseInt(reader.readLine());
    // 入力が0以外なら繰り返す
    while (input != 0) {
      System.out.println("Hello!");
      // 次の入力
      System.out.print("0で終了:");
      input = Integer.parseInt(reader.readLine());
    }
  }
}
```

最初の入力で input に 0 が入るとwhileループは一度も実行されない

```
public class UntilLoop {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    // 入力を保存する変数を宣言
    int input;
    // 最初の入力
    System.out.print("0で終了:");
    input = Integer.parseInt(reader.readLine());
    // 入力が0以外なら繰り返す
    while (input != 0) {
      System.out.println("Hello!");
      // 次の入力
      System.out.print("0で終了:");
      input = Integer.parseInt(reader.readLine());
    }
  }
}
```

最初の入力で input に 0 以外が入るとwhileループの第1回が実行される

```
public class UntilLoop {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    // 入力を保存する変数を宣言
    int input;
    // 最初の入力
    System.out.print("0で終了:");
    input = Integer.parseInt(reader.readLine());
    // 入力が0以外なら繰り返す
    while (input != 0) {
      System.out.println("Hello!");
      // 次の入力
      System.out.print("0で終了:");
      input = Integer.parseInt(reader.readLine());
    }
  }
}
```

ループ内の入力で input に 0 以外が入るとwhileループの次の回に進める

```
public class UntilLoop {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    // 入力を保存する変数を宣言
    int input;
    // 最初の入力
    System.out.print("0で終了:");
    input = Integer.parseInt(reader.readLine());
    // 入力が0以外なら繰り返す
    while (input != 0) {
      System.out.println("Hello!");
      // 次の入力
      System.out.print("0で終了:");
      input = Integer.parseInt(reader.readLine());
    }
  }
}
```

ループ内の入力で input に 0 が入るとwhileループは打ち切られる

```
public class UntilLoop {
  public static void main(String[] args) throws IOException {
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(System.in));
    // 入力を保存する変数を宣言
    int input;
    // 最初の入力
    System.out.print("0で終了:");
    input = Integer.parseInt(reader.readLine());
    // 入力が0以外なら繰り返す
    while (input != 0) {
      System.out.println("Hello!");
      // 次の入力
      System.out.print("0で終了:");
      input = Integer.parseInt(reader.readLine());
    }
  }
}
```

課題

各自のペースで
「第05週目の課題」をやってみよう

TriangleCheck のヒント

Javaの文法通りではないので注意

```
if (a = b = c)
    正三角形
else if (a = b または b = c または c = a)
    二等辺三角形
else
    一般の三角形
```

ScoreAverage のヒント

入力データ格納のための変数、合計のための変数、
データ個数数えるための変数などの宣言初期化

最初の入力

```
if (入力<0 || 入力>100)
   入力データなしのときの処理
else{
   while ( 継続条件 ) {
      入力データを合計に加える
      データ個数を1増やす
      入力する
   }
   合計をデータ個数で割って出力
}
```