# 情報処理技法
# (Javaプログラミング)1

## 第13回
## ファイルとやりとりするには?

人間科学科コミュニケーション専攻
白銀 純子

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 第13回の内容

- ファイル入出力

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 前回の復習問題の解答

- 下記の計算で、桁落ちをせずに計算をさせるには、どのように修正すれば良いかを答えなさい。
  - 変数result: double型の変数
  - 変数sum: int型の変数

```
result = sum / 7;
```

解答: result = (double) / 7;

- 4つの単語java、program、file、nameをこの順序でつなげて変数名としたいとき、どのような変数名にするのが望ましいかを答えなさい。

解答: javaProgramFileName

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 第2回課題その4(1)

- 乱数をArrayListに登録するには?(間違い例)

randomのデータを決定しているのは、for文に入る前
= randomの中のデータは、for文の中でもずっと同じ
= randomListに登録されるデータは全て同じ

```
int i, random;
ArrayList<Integer> randomList = new ArrayList<Integer>();

random = (int) Math.random() * 10000;
for(i = 1; i <= 10000; i = i + 1) {
  randomList.add(random);
}
```

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 第2回課題その4(2)

- 乱数をArrayListに登録するには?(正解例)

1つ乱数を作る→ArrayListに登録を繰り返す

```
int i, random;
ArrayList<Integer> randomList = new ArrayList<Integer>();
for (i = 1; i <= 10000; i = i + 1) {
  random = (int) (Math.random() * 10) + 1;
  randomList.add(random);
}
```

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 第2回課題その4(3)

- 入力された数が、ArrayList内に何個登録されているかを数えるには?
  1. 個数を表す変数を1つ用意して、0で初期化しておく
  2. ArrayListに登録されている数を1つ1つ取り出して入力された数と同じかどうか比較する
  3. 2. の処理で、同じであれば、1. で用意した変数の値を1増やす
     - バードウォッチングなどでカチカチとカウンターでカウントするイメージ

```
int count = 0;
for (int rand: randomList) {
    if (rand == num) {
        count = count + 1;
    }
}
```

- randomList: 乱数を登録したArrayList
- num: 標準入力で入力された数

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ファイル入出力

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ファイル入出力とは?(p. 257)

- ファイル入力: ファイルに書き込まれている内容をプログラム中に読み込むこと
  - 標準入力: ターミナルに書き込まれた内容をプログラム中に読み込むこと
- ファイル出力: プログラムの実行結果などをファイルの中に書き込むこと
  - 標準出力: プログラムの実行結果をターミナルに書き込むこと

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ファイル入力

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～入力～(全体像)(p. 258)

```
try {
   String str;

   FileReader  fr = new  FileReader(入力するファイル名);
   BufferedReader  br = new  BufferedReader(fr);

   str = br.readLine();
   br.close();
   fr.close();
 }
catch (IOException e) {
    System.out.println("指定されたファイルの入力ができません。");
 }
```

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～入力～(詳細1)(p. 259)

```
try {
   String str;

   FileReader  fr = new  FileReade
   BufferedReader  br = new  Buff

   str = br.readLine();
   br.close();
   fr.close();
 }
catch (IOException e) {
      System.out.println("指定されたファイルの入力ができません。");
 }
```

ファイル入力では、
- 入力すべきファイルが存在しない
- ファイルを読む権限がない

などのエラーが起こることもある

例外処理が必要

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～入力～(詳細2)(p. 260)

```
try {
   String str;

   FileReader  fr = new  FileReader(入力するファイル名);
   BufferedReader  br = new  BufferedReader(fr);

   str = br.readLine();
   br.close();
   fr.close();
 }
catch (IOException e) {
        System.out.println("指定されたファイルの入力ができません。");
 }
```

どのファイルの内容を入力するか、ファイルの名前を決める

- 「入力するファイル名」は、単なる文字列でも、変数に代入された文字列でも良い
- 入力するファイルは、Javaプログラムと同じ場所に置いておく

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～入力～(詳細3)(p. 260)

```
try {
   String str;

   FileReader  fr = new  FileReader(入力するファイル名);
   BufferedReader  br = new  BufferedReader(fr);

   str = br.readLine();
   br.close();
   fr.close();
 }
catch (IOException e) {
        System.out.println("指定されたファイルの入力ができません。");
 }
```

ファイルの内容に対する操作には、
- 内容を読む
- 内容を書き込む

の2種類がある。1度に行うことのできる操作はどちらか一方なので、「読む」のか「書く」のかを決めておく

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～入力～(詳細4)(p. 260)

```
try {
   String str;

   FileReader  fr = new  FileReader(入力するファイル名);
   BufferedReader  br = new  BufferedReader(fr);

   str = br.readLine();
   br.close();
   fr.close();
 }
catch (IOException e) {
        System.out.println("指定されたファイルの入力ができません。");
 }
```

「読むために」ファイルを開く作業(「fr」の部分は、FileReaderの変数名を入れること)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～入力～(詳細5)(p. 261)

```
try {
   String str;

   FileReader  fr = new  FileReader(入力するファイル名);
   BufferedReader  br = new  BufferedReader(fr);

   str = br.readLine();
   br.close();
   fr.close();
 }
catch (IOException e) {
        System.out.println("指定
 }
```

開いたファイルを1行ずつ(文字列として)読んでいく(「br」の部分は、BufferedReaderの変数名を入れること)

- 1つ目の「br.readLine()」で1行目を読む
- 2つ目の「br.readLine()」で2行目を読む
- .....

※ただし、2行目を読んだあとに1行目をもう一度読んだり、2行目をとばして3行目を読む、ということはできない

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～入力～(詳細6)(p. 261)

```
try {
   String str;

   FileReader  fr = new  FileReader(入力するファイル名);
   BufferedReader  br = new  BufferedReader(fr);

   str = br.readLine();
   br.close();
   fr.close();
 }
catch (IOException e) {
        System.out.println("
 }
```

ファイルを読み終わった後、ファイルを閉じる作業
(プログラムでは、開いたファイルは必ず閉じる
作業をしなければならない)
(「br」の部分は、BufferedReaderの変数名、
「fr」の部分はFileReaderの変数名を入れること)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ファイル入力の例(p. 262)

プログラム:

```
try {
   String str;

   FileReader fr = new FileReader("Sample.txt");
   BufferedReader br = new BufferedReader(fr);

   str = br.readLine();
   System.out.println(str);
   br.close();
   fr.close();
 }
catch (IOException e) {
        System.out.println("指定されたファイルの入力ができません。");
 }
```

「Sample.txt」の中身:

Javaプログラミング
サンプルファイル

出力結果:

Javaプログラミング

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 何行も書かれているファイル(p. 267)

- ファイルを最初から最後まで全部読むには?
  - 「br.readLine()」をファイルの行数分書く?
  - ファイルの行数が多いときは?
  - ファイルの行数が何行かわからないときは?

「ファイルが読み込み可能」
という条件でwhile文を使う

ファイルの最後の行まで読み込んでしまうと、その次の行を
読み込もうとしたときに「読み込み不可能」という結果が返ってくる

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 「読み込み可能」という条件[1](p. 267)

```
String str;
while (br.ready()) {
   読み込んだ文字列に対する処理
}
```

- 「br.ready()」で、ファイルの中にまだ読み込んでいない行が存在するかをチェック
  - ✓ 読み込んでいない行が存在すれば、「true」という結果
  - ✓ 読み込んでいない行が存在しなければ、「false」という結果
    - ➡ ファイルの終わりに達していない限り、whileの中身を実行する

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 「読み込み可能」という条件[2](p. 267)

```
try {
   String str;

   FileReader fr = new FileReader("Sample.txt");
   BufferedReader br = new BufferedReader(fr);

   while (br.ready()) {
      str = br.readLine();
      …….
   }
   br.close();
   fr.close();
}
catch (IOException e) {
       System.out.println("指定されたファイルの入力ができません。");
}
```

ファイルを開く

ファイルの終わりに達するまで各行を読み込み、読み込んだ各行の処理をする

読み込み終わったらファイルを閉じる

- ファイルを閉じると、それ以上は読み込みできない

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# while文の中身は...(p. 267)

- while文の中身(読み込んだ1行1行の扱い)は、プログラムの目的に応じて書く
  - strの内容を配列に代入する
    →ファイルの1行1行を配列として扱う
  - strの内容を数値に変換する(そして配列に代入する)
  - strの内容を細かく分解する(例えばスペースで区切って単語に分解する)
  - etc.

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ファイル入力の例外処理(1)

- FileNotFoundException
  - ファイルに関する例外
    - 読み込もうとしたファイルが存在しない場合
  - 分類されているパッケージ: java.io
    - 「import java.io.FileNotFoundException」または「import java.io.*;」がなければコンパイルエラー
  - ただし、IOExceptionを使っていれば、FileNotFoundExceptionは不要
    - IOExceptionは、FileNotFoundExceptionも兼ねている
    - ファイルは存在しても読み書きできないのか、ファイルが存在自体しないのか、を区別したいなどのときには両方利用する

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ファイル入力の例外処理(2)

```
String str;
try {
   FileReader  fr = new  FileReader(入力するファイル名);
   BufferedReader  br = new  BufferedReader(fr);

   str = br.readLine();
   br.close();
}
catch (FileNotFoundException e) {
}
catch(IOException e) {
}
```

例外が発生する可能性のあるポイント
(ファイルの読み込みを決めているポイント)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ファイル入力の例外処理(3)

- IOExceptionとFileNotFoundExceptionの関係
  - IOExceptionはFileNotFoundExceptionを兼ねている
    - IOExceptionをFileNotFoundExceptionの前に書くと、FileNotFoundExceptionの例外が発生することはない

コンパイルエラー

FileNotFoundExceptionは、IOExceptionの前に書く必要
- ファイルが読み込めない理由を限定しなくて良い場合はIOExceptionだけで良い
- ファイルが読み込めない理由をある程度限定したい場合は両方とも書く

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ループ文利用の注意(入力)

```
try {
   String str;

   FileReader fr = new FileReader("Sample.txt");
   BufferedReader br = new BufferedReader(fr);

   while (br.ready()) {
      str = br.readLine();
      ……
      br.close();
      fr.close();
   }
}
catch (IOException e) {
        System.out.println("ファイルの入力ができません。");
}
```

ループ文の中に、ファイルを閉じる処理が入っている

while文の処理に入ったときに…

1. 「br.ready()」(条件)はtrueなのでループ文の処理開始
2. ファイルの1行目を読み込む
3. 1行目に対する処理を行う
4. ファイルを閉じる
5. 「br.ready()」(条件)がfalseなので、ループ文終了
   (ファイル読み込み終了)
   - 1行目の読み込みのときにファイルを閉じているから

ファイルの2行目以降は読み込み・処理されない

ファイルを閉じる処理はループ文の後に!

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ファイル出力

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～出力～(全体像)(p. 272)

```
try {
   FileWriter  fw = new  FileWriter(出力するファイルの名前);
   BufferedWriter  bw = new  BufferedWriter(fw);
   PrintWriter  pw = new  PrintWriter(bw);

   pw.println(書き込む内容);
   pw.close();
   bw.close();
   fw.close();
 }
catch (IOException e) {
        System.out.println("指定されたファイルに出力ができません。");
 }
```

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～出力～(詳細1)(p. 273)

```
try {
   String str;
   FileWriter  fw = new  FileWrite
   BufferedWriter  bw = new  Buff
   PrintWriter  pw = new  PrintWri

   pw.println(書き込む内容);
   pw.close();
   bw.close();
   fw.close();
}
catch (IOException e) {
      System.out.println("指定されたファイルに出力ができません。");
}
```

ファイル出力では、
- フォルダにファイルを作成できない
- ファイルに書き込む権限がない

などのエラーが起こることもある

↓

例外処理が必要

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～出力～(詳細2)(p. 273)

```
try {
   String str;
   FileWriter  fw = new  FileWriter(出力するファイルの名前);
   BufferedWriter  bw = new  BufferedWriter(fw);
   PrintWriter  pw = new
                                    どのファイルに書き込むか、ファイルの名前を決める

   pw.println(書き込む内容);
   pw.close();
   bw.close();
   fw.close();
 }
catch (IOException e) {
        System.out.println("指定されたファイルに出力ができません。");
 }
```

- 「出力するファイル名」は、単なる文字列でも、変数に代入された文字列でも良い
- 出力するファイルは、Javaプログラムと同じ場所に作成される

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～出力～(詳細3)(p. 273)

```
try {
   String str;
   FileWriter  fw = new  FileWriter(出力するファイルの名前);
   BufferedWriter  bw = new  BufferedWriter(fw);
   PrintWriter  pw = new  PrintWriter(bw);

   pw.println(書き
   pw.close();
   bw.close();
   fw.close();
 }
catch (IOException e) {
        System.out.println("指定されたファイルに出力ができません。");
 }
```

ファイルの内容に対する操作には、
- 内容を読む
- 内容を書き込む

の2種類がある。1度に行うことのできる操作はどちらか一方なので、「読む」のか「書く」のかを決めておく

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書き込むファイル(p. 283)

- 「FileWriter fw = new FileWriter(出力するファイルの名前);」で指定されたファイルがすでに存在する場合:
  →ファイルは一旦削除され、新しいファイルが作成
  (もともとのファイルの内容が消去される)
- すでに存在するファイルの最後に追加して書き込むことは可能

  「FileWriter fw = new FileWriter(出力するファイルの名前, true)」とする

- すでに存在するファイルの途中に追加は不可能
  - ファイルを全て読み込み、必要な部分を追加して書き込むという対応

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～出力～(詳細4)(p. 273)

```
try {
   String str;
   FileWriter  fw = new  FileWriter(出力するファイルの名前);
   BufferedWriter  bw = new  BufferedWriter(fw);
   PrintWriter  pw = new  PrintWriter(bw);

   pw.println(書き込む内容);
   pw.close();
   bw.close();
   fw.close();
 }
catch (IOException e) {
        System.out.println("指定されたファイルに出力ができません。");
 }
```

「書き込むために」ファイルを開く作業(「fw」の部分はFileWriterの変数名、「bw」の部分はBufferedWriterの変数名を入れること)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～出力～(詳細5)(p. 276)

```
try {
   String str;
   FileWriter  fw = new  FileWriter(出力するファイルの名前);
   BufferedWriter  bw = new  BufferedWriter(fw);
   PrintWriter  pw = new  PrintWriter(bw);

   pw.println(書き込む内容);
   pw.close();
   bw.close();
   fw.close();
 }
catch (IOExceptio
        System.out.p
 }
```

開いたファイルに内容を書き込んでいく(「pw」の部分は、PrintWriterの変数名を入れること)

- 「pw.println(...)」では、書き込んだ後に改行が入る
- 「pw.print(...)」では、書き込んだ後に改行が入らない

→目的に応じて使い分けること

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 書式～出力～(詳細6)(p. 278)

```
try {
   String str;
   FileWriter  fw = new  FileWriter(出力するファイルの名前);
   BufferedWriter  bw = new  BufferedWriter(fw);
   PrintWriter  pw = new  PrintWriter(bw);

   pw.println(書き込む内容);
   pw.close();
   bw.close();
   fw.close();
 }
catch (IOException e) {
        System.out.println("
 }
```

ファイル書き込みが終わった後、ファイルを閉じる作業
(プログラムでは、開いたファイルは必ず閉じる
作業をしなければならない)
(「pw」・「bw」・「fw」の部分は、それぞれPrintWriter・
BufferedWriter・FileWriterの変数名を入れること)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ファイルを閉じる順序の注意

```
try {
   String str;
   FileWriter  fw = new
   BufferedWriter  bw =
   PrintWriter  pw = new

   pw.println(書き込む内
   pw.close();
   bw.close();
   fw.close();
 }
catch (IOException e) {
 }
```

必ず

1. PrintWriter
2. BufferedWriter
3. FileWiter

の順(ファイル出力の準備をしたのとは逆の順序)で閉じること

- 書き込み内容は、一旦バッファという一時保存領域に保存されている
- PrintWriterを閉じたときに、バッファの内容がファイルに書き込まれる
- PrintWriter・BufferedWriter・FileWriterの閉じる順序を間違えると、ファイル書き込み前にバッファの内容が消去される
  = ファイルに内容が書き込まれない

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ファイル出力の例(p. 280)

プログラム:

```
try {
   String str;
   FileWriter fw = new FileWriter("Sample.txt");
   BufferedWriter  bw = new  BufferedWriter(fw);
   PrintWriter pw = new PrintWriter(fw);

   pw.println("Javaプログラミング");
   pw.print("サンプルファイル");
   pw.close();
   bw.close();
   fw.close();
 }
catch (IOException e) {
        System.out.println("指定されたファイルに出力ができません。");
 }
```

実行結果
「Sample.txt」の中身:

Javaプログラミング
サンプルファイル

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# ループ文利用の注意(出力)

```
try {
   FileWriter fw = new FileWriter("Sample.txt");
   BufferedWriter  bw = new  BufferedWriter(fw);
   PrintWriter pw = new PrintWriter(fw);

   int i;
   for (i = 0; i < 10; i = i + 1) {
      .........
      pw.println(i + "番目");
      pw.close();
      bw.close();
      fw.close();
   }
}
catch (IOException e) {
       System.out.println("ファイルの出力ができません。");
} ..........
```

ループ文の中に、ファイルを閉じる処理が入っている

for文の処理に入ったときに…

1. 「i < 10」(条件)はtrueなのでループ文の処理開始
2. 必要な処理を行う
3. ファイルの1行目の書き込みをする
4. ファイルを閉じる
5. 「i < 10」(条件)がtrueなので、ループ文の処理継続
6. 必要な処理を行う
7. IOExceptionの例外が発生する
   - for文の1回目の処理のときにファイルを閉じているため

ファイルの2行目以降は書き込みされない

ファイルを閉じる処理はループ文の後に!

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 様々な入出力をする場合(1)

- 1つのプログラム内で様々な入出力をする場合のtry～catch

- 1つにまとめて良い
  - ✓ ただし、標準入力・ファイル入出力のどれで例外が発生したかは把握しづらくなる
- それぞれ別個にtry～catchを設けても良い
  - ✓ 変数宣言の位置には注意が必要

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 様々な入出力をする場合(2)

- 1つにまとめる場合(1)

```
try {
   BufferedReader standardBr = new BufferedReader(new InputStreamReader(System.in));
   String str = standardBr.readLine();

   FileReader fr = new FileReader("TestFile.txt");
   BufferedReader fileBr = new BufferedReader(fr);

   String line;
   while (fileBr.ready()) {
      line = fileBr.readLine();
   }

   FileWriter fw = new FileWriter("Sample.txt");
   BufferedWriter  bw = new  BufferedWriter(fw);
   PrintWriter pw = new PrintWriter(fw);

   .....
}
catch (IOException e) {
 }
```

try～catchは1つだけ

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 様々な入出力をする場合(3)

- 1つにまとめる場合(2)

```
try {
  BufferedReader standardBr = new BufferedReader(new InputStreamReader(System.in));
  String str = standardBr.readLine();

  FileReader fr = new FileReader("TestFile.txt");
  BufferedReader fileBr = new BufferedReader(fr);

  String line;
  while (fileBr.ready()) {
    line = fileBr.readLine();
  }

  FileWriter fw = new FileWriter("Sample.txt");
  BufferedWriter  bw = new  BufferedWriter(fw);
  PrintWriter pw = new PrintWriter(fw);

  .....
}
catch (IOExc
 }
```

入力の時のBufferedReaderの「br」や「reader」は変数

- try～catchをまとめる場合は、標準入力とファイル入力で違う変数にすること

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 様々な入出力をする場合(4)

- 別個に設ける場合(1)

```
String str, line;
try {
   BufferedReader standardBr = new BufferedReader(new InputStreamReader(System.in));
   str = standardBr.readLine();
   .....
}
catch(IOException e) {
}
try {
   FileReader fr = new FileReader("TestFile.txt");
   BufferedReader fileBr = new BufferedReader(fr);
   while (fileBr.ready()) {
      line = fileBr.readLine();
   }
}
catch(IOException e) {
}
try {
   FileWriter fw = new FileWriter("Sample.txt");
   BufferedWriter  bw = new  BufferedWriter(fw);
   PrintWriter pw = new PrintWriter(fw);
   .....
}
catch (IOException e) {
}
```

それぞれにtry～catchを記述

Copyright (C) Junko S

# 様々な入出力をする場合(5)

- 別個に設ける場合(2)

```
String str, line;
try {
   BufferedReader standardBr = new BufferedReader(new InputStreamReader(System.in));
   str = standardBr.readLine();
   .....
}
catch(IOException e) {
}
   }
}
catch(IOException e) {
}
try {
   FileWriter fw = new FileWriter("Sample.txt");
   BufferedWriter  bw = new  BufferedWriter(fw);
   PrintWriter pw = new PrintWriter(fw);
   .....
}
catch (IOException e) {
}
```

- 変数宣言は、try～catchよりも前に!
  - ✓ 変数は宣言をした「{」～「}」の中でしか使えない
    = 1つ目のtryの中で宣言された変数は、2つ目以降のtry～catchでは使えない(コンパイルエラー)

Copyright (C) Junko S

# やってみよう!

- 教科書p. 286の例題01-07をやってみよう

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016, All rights reserved.

# 期末試験

- 期末試験
  - 7月26日(火) 2限 24102教室
  - 90分
  - 持ち込みすべて可の実技メインの試験
    - 筆記も少しあり