教科書「オブジェクト指向　Javaプログラミング入門」

- 版（刷）の違いによる内容の変更点
  - 2020年2月29日発行の第2版初版第2刷 P215 にソースコード10.4掲載なし
  - 2021年3月31日発行の第2版初版第3刷 P215 にソースコード10.4掲載あり

- ソースコード10.4とは？

  try-with-resources 文を使わないファイル入力のサンプルコード

KINDAI UNIVERSITY

KDIX
Kindai University Digital Informatics eXperts

リソースを利用するコードは，

・ファイル
・データベース　etc.

- **リソースが不要になった時点で，それを開放する処理が必要**
  ファイル入出力処理の場合は，読み／書き用に開いたファイルをクローズする作業
- **リソースを適切に開放することで，意図しないバグを防ぐ**

ファイル（リソース）入出力処理におけるリソースの解放方法

・try-with-resource文を使う　（教科書ソースコード10.1）
・try-catch-finally文を使う　（教科書ソースコード10.4）
・その他

ソースコード10.4の訂正

ソースコード 10.4　このままコンパイルすると**エラー**になる

```
 1～7 行目は ソースコード 10.1 と同じ
 8          BufferedReader reader;
 9          try {
10                reader = Files.newBufferedReader(path);
11                String line;
12                while ((line = reader.readLine()) != null) {
13                   System.out.printf("%s\n", line);
14                }
15          } catch (IOException e) {
16             System.out.println(e);
17          } finally {
18             if (reader != null) {
19                reader.close();
20             }
21          }
22    }
```

置き換え

```
BufferedReader reader = null;
```

置き換え

この部分を

```
try{                                  //追加
   if (reader != null){
      reader.close();
   }
}  catch(IOException e){   //追加
    System.out.printf(e); //追加
}                                     //追加
```

close（）メソッドにも例外処理がないと
コンパイルエラーになる

ソースコード10.1

```
import java.nio.file.*;
import java.io.BufferedReader;
import java.io.IOException;

public class FileReadSample {
  public static void main(String[] args){
    Path path = Paths.get(args[0]);

    try (BufferedReader reader = Files.newBufferedReader(path)){
        String line;
        while ((line = reader.readLine()) != null){
           System.out.printf("%s¥n", line);
        }
    }  catch (IOException e){
         System.out.println(e);
    }
  }
}
```

readerのスコープ

try-with-resources 文によるファイル入力

利点：try(リソースの生成処理){}と記述するだけで，tryブロックが終了する際に暗黙的にリソースの解放（ファイルクローズ）処理を行ってくれる．

欠点：BufferedReader型オブジェクトreaderのスコープは，tryブロック内だけ．tryの外でreaderは参照できない．

ソースコード10.4

```
import java.nio.file.*;
import java.io.BufferedReader;
import java.io.IOException;

public class FileReadSample_10_4 {
   public static void main(String[] args){
      Path path = Paths.get(args[0]);
      BufferedReader reader = null;
      try {
         reader = Files.newBufferedReader(path);
         String line;
         while ((line = reader.readLine()) != null){
            System.out.printf("%s¥n", line);
         }
      }  catch (IOException e){
            System.out.println(e);
      }  finally{
            try{
               if (reader != null){
                  reader.close();
               }
            }  catch(IOException e){
               System.out.printf(e);
            }
      }
   }
}
```

readerのスコープ

try{}の外でreaderを定義

readerのスコープ内なので，finally{}の中で reader.close(); しても，コンパイルエラーにならない．

try-catch-finally 文によるファイル入力

利点：try{}の外でリソースを参照できる．

欠点：close()を明示的に書かなければいけない．
↑忘れると，アブナイ．．．