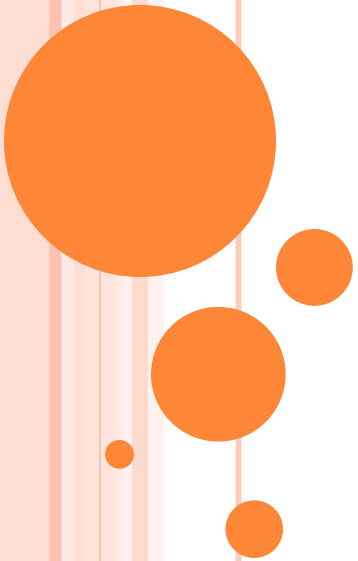


File IO

オブジェクト指向プログラミング特論

只木進一：総合情報基盤センター

JavaでのFile IO

- JavaでのFile IOの仕組み
  - 言語(java.langパッケージ)にはFile IOが含まれない
  - 標準入出力のみ
  - java.ioパッケージが別に用意されている
- 例外処理の実際
  - IOでは、エラーが発生しやすい
  - 読めない、書けない
  - ファイルが存在しない

標準入出力

```
package java.lang;
import java.io.*;
public final class System {
  private System() {}//インスタンスは作成不能
  public final static InputStream in;
  public final static PrintStream out;
  public final static PrintStream err;
...
}
```

標準入力:キーボード

- 一文字ずつの入力
  - メソッドread()を使用
- 戻り値
  - 正整数:文字
  - -1:終了
- 例外発生可能性
  - IOException

```
StringBuilder b=new StringBuilder();
int c;
try {
    while ((c = System.in.read()) != -1) {
        b.append((char)c);
        //1バイトずつ読んでbに追加
   }
} catch (IOException ex) {
  //エラー処理
}
```

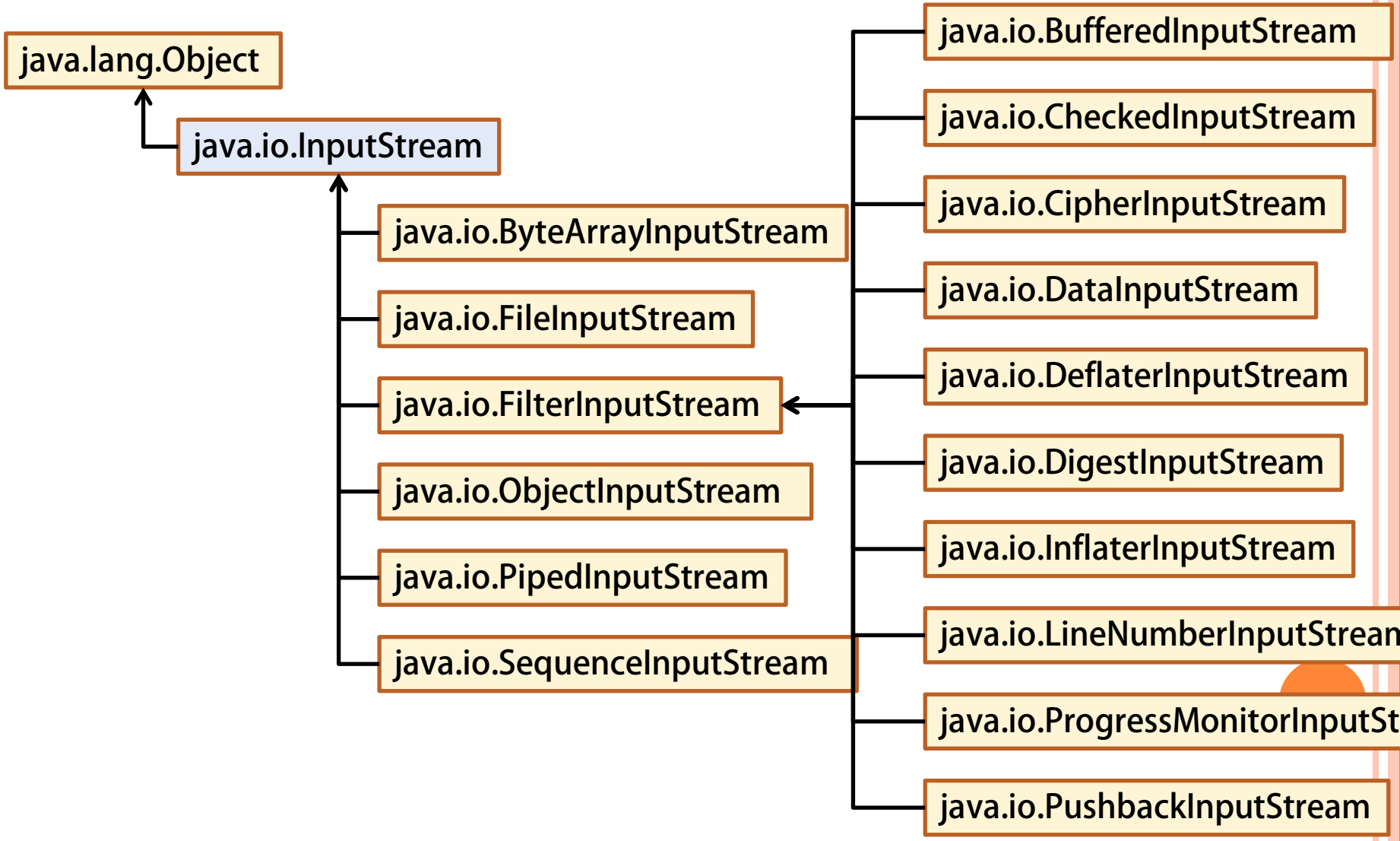
標準出力：端末へ

- メソッドprint()：改行なし
- メソッドprintln()：改行あり
- 引数
  - 原始型
  - オブジェクト
    - toString()メソッドを使うなどして、文字列に変換
    - Object.toString()

java.io.File

- ファイルそのもののクラス
- ファイルの属性を調べることができる
  - 存在するか否か
  - read/writeパーミッション
  - ファイルかディレクトリ
- ファイルの生成、消去が可能
- インスタンス生成時には、ファイルにアクセスしていないことに注意

入力ストリームのクラス階層



InputStreamの基本:1

- 読み込みはbyte単位
  - int read();//1 byte読み込み
  - int read(byte[]); //byte[]で一括読み込み
- 閉鎖
  - void close();

InputStreamの基本:2

- ファイルからの読み込み

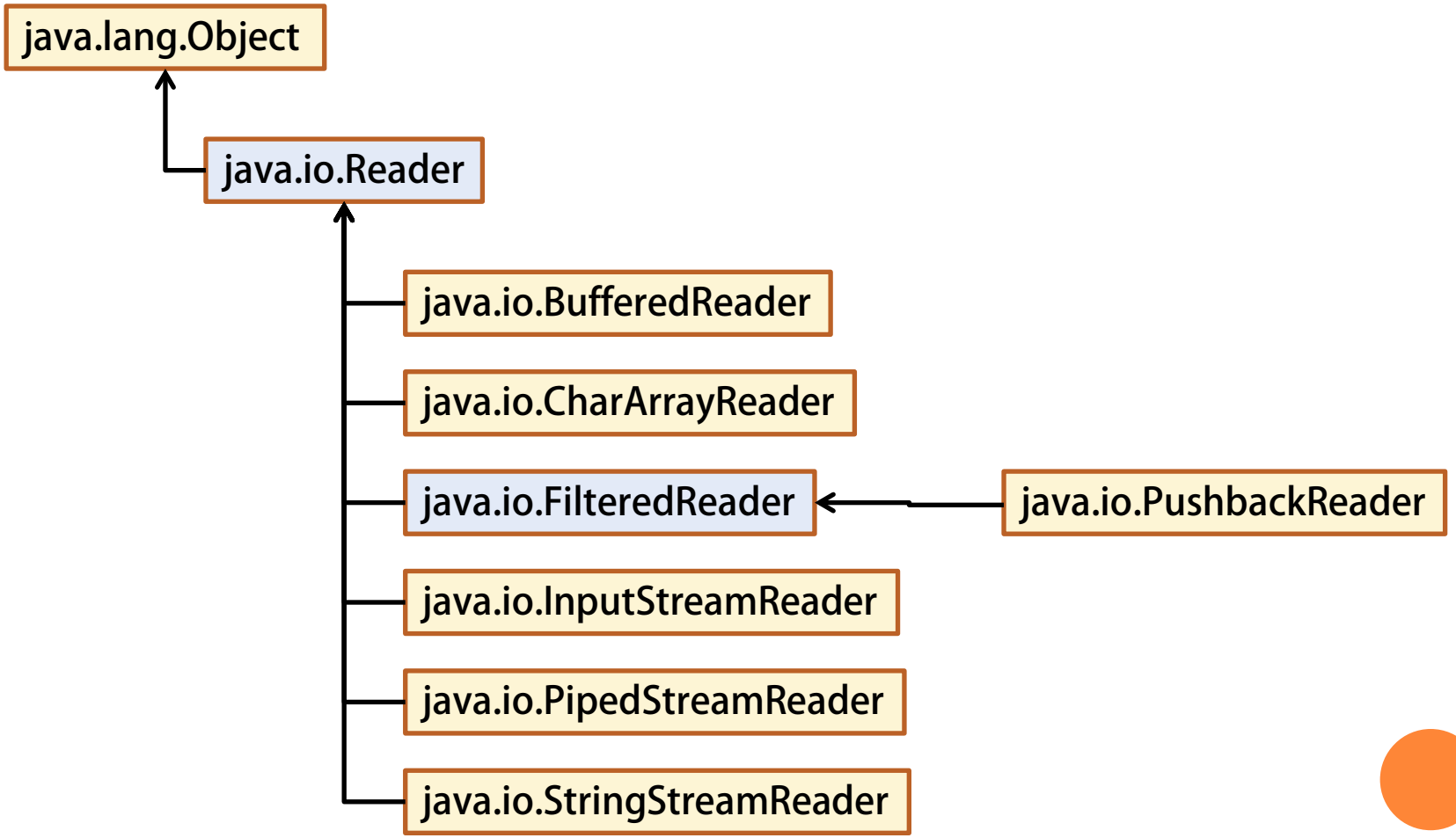
```
File file = new File("input.txt");//ファイル指定
BufferedInputStream in=null;
try {
    //入力バッファを開く
    in = new BufferedInputStream(new FileInputStream(file));
} catch (FileNotFoundException ex) {//ファイルが開けられない場合
    System.err.println(ex);
}

int n;
while((n=in.read())!=-1){//1バイト毎に読み込み
    System.out.print((char)n);
}
```

READERを使う

- 文字、文字列単位での読み込み
  - int read(); //一文字読み込み
  - int read(char[]);//文字配列へ読み込み
  - String readLine();//一行を文字列へ読み込み

Readerのクラス階層



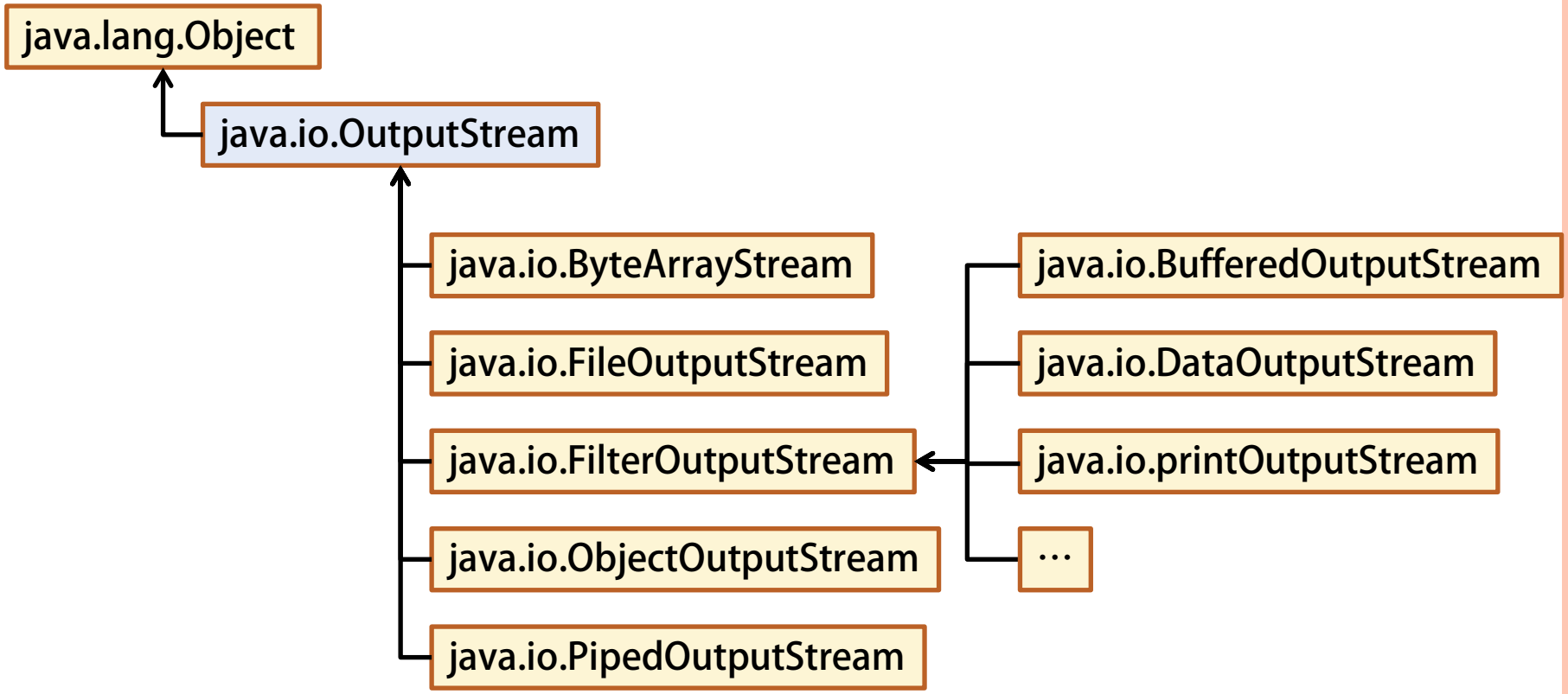
```
File file = new File("input.txt");
        BufferedReader in = null;
        try {
            in = new BufferedReader(
                        new InputStreamReader(new FileInputStream(file)));
        } catch (FileNotFoundException ex) {
            System.err.println(ex);
        }
        try {
            String line;
            while ((line = in.readLine()) != null) {
                System.out.print(line);
            }
            in.close();
        } catch (IOException ex) {
            System.err.println(ex);
        }
```

標準入力のWRAPPING

- 標準入力をBufferedReaderの一種に見せかける
  - ファイル書き込みと標準出力を切り替えることができる

```
BufferedReader in
    = new BufferedReader(new InputStreamReader(System.in));
try {
    String line;
    while ((line = in.readLine()) != null) {
        System.out.print(line);
    }
    in.close();
} catch (IOException ex) {
    System.err.println(ex);
}
```

出力ストリームのクラス階層



OutputStreamの基本

- バイト単位の書き出し
  - void write(byte[]);
- flush：強制排出
  - void flush();
- 閉鎖
  - void close();

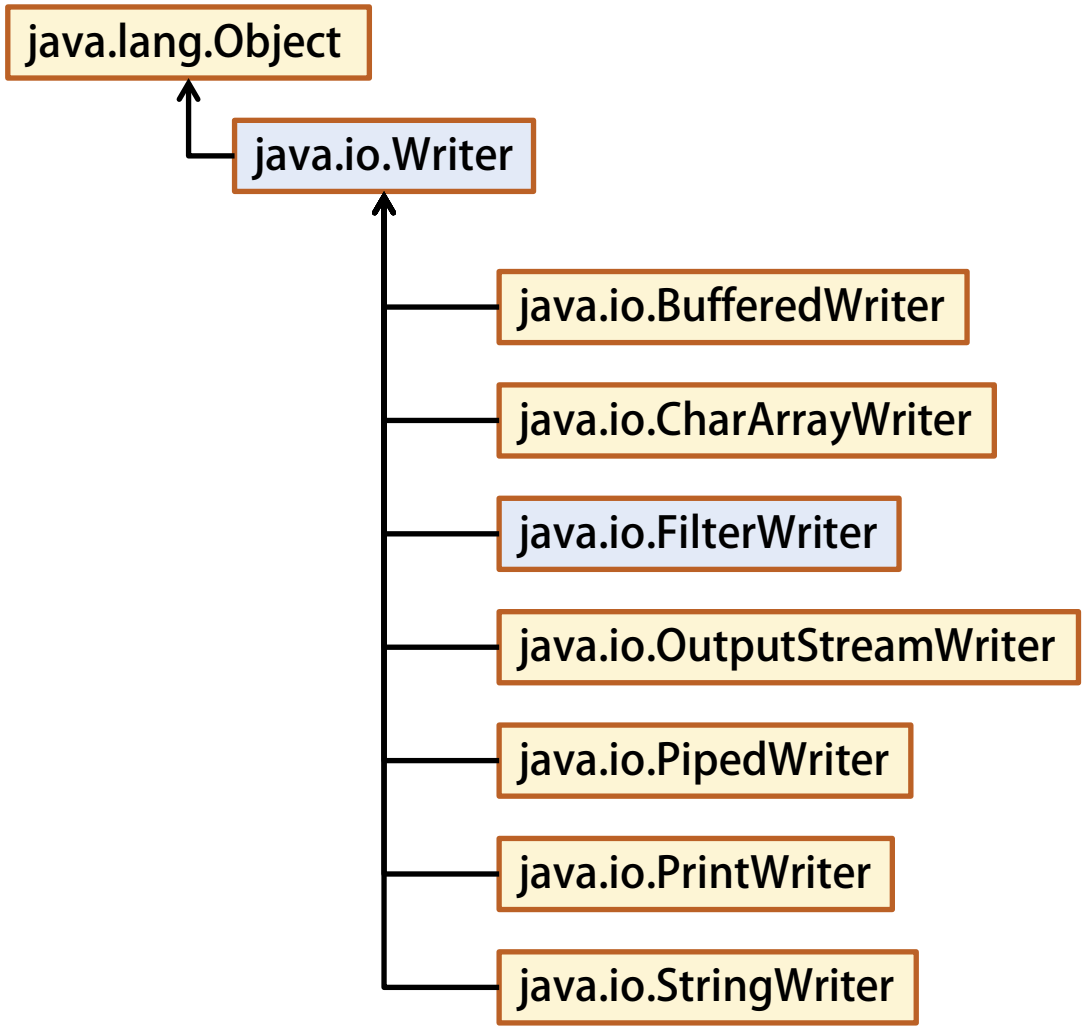
PrintStream

- OutputStreamに機能を追加
  - 文字列書き出し
  - print(String);
  - print(Object)
    - //Object.toStream()が使用される
  - println(String)
  - println(Object)
- 一文字追加
  - append(char);

Writer

- 文字、文字列をストリームに書く
  - void write(char);
  - void write(String);

WRITERのクラス階層



Writerの例

```
BufferedReader in;
BufferedWriter out;
try {
    in = new BufferedReader(
        new InputStreamReader(new FileInputStream(inFile)));
    out = new BufferedWriter(
        new OutputStreamWriter(new FileOutputStream(outFile)));
} catch (FileNotFoundException ex) {System.err.println(ex);}
try {
    String line;
    while ((line = in.readLine()) != null) {
        out.write(line);
        out.newline()l//改行
}
in.close();
out.close();
} catch (IOException ex) {System.err.println(ex);}
```

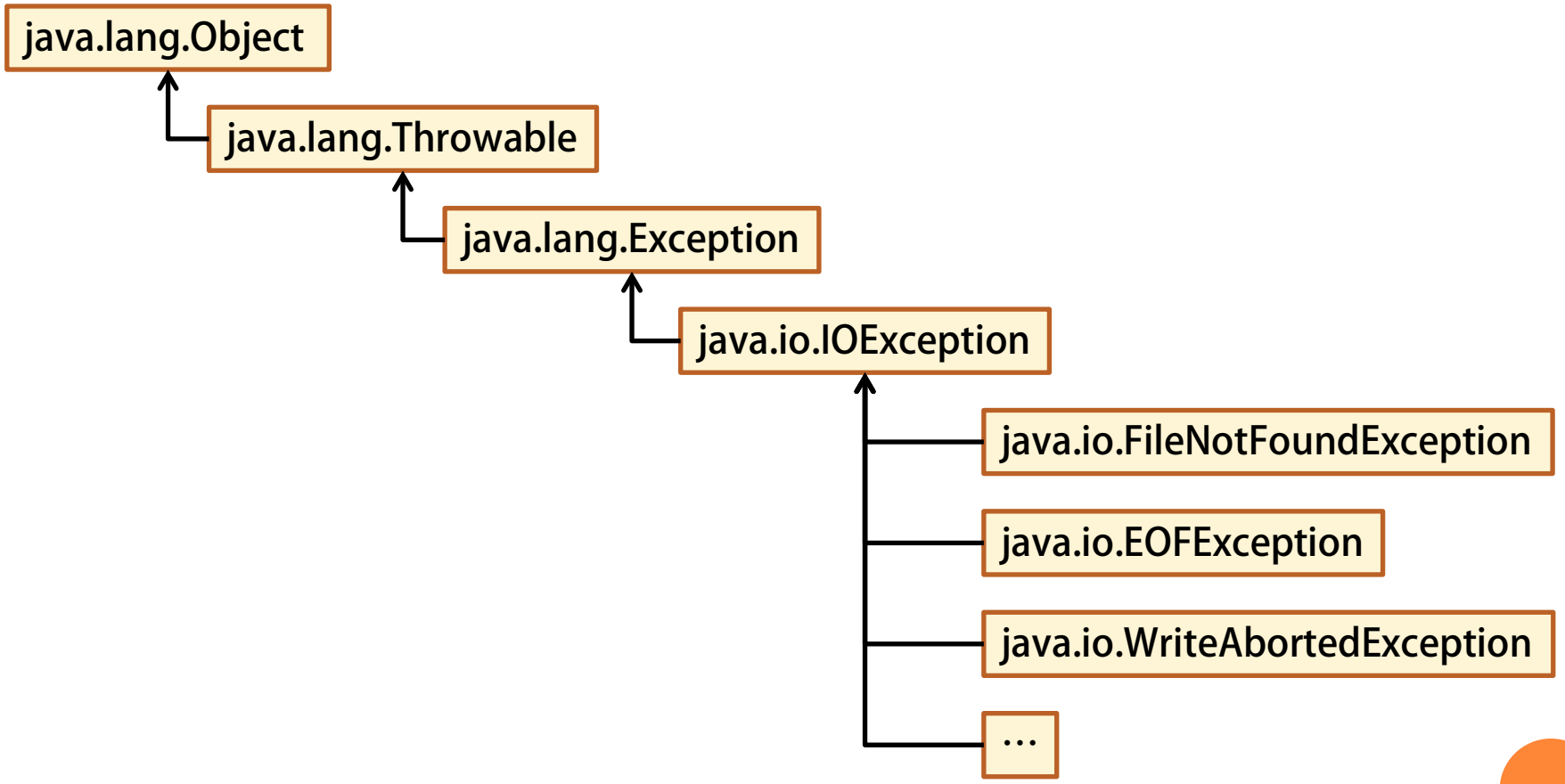
標準出力のWRAPPING

```
BufferedWriter out
        = new BufferedWriter(new OutputStreamWriter(System.out));
String nl=System.getProperty("line.separator");
try{
    out.write(“Something”);
    out.write(nl);
}catch(IOException ex){
}
```

例外処理

- ファイルの入出力では、実行時エラーが発生
- ファイルが読めない、ファイルに書けない
- メソッド間での例外処理の方法の統一が必要
  - ライブラリとしての挙動の統一
  - ユーザプログラムでの例外処理の簡素化
- 例外もクラスとして定義する

IOExceptionのクラス階層



例外を捕まえる

- 例外を発生させるメソッドの実行
  - tryブロックで囲む
- 例外時の処理
  - 例外をcatchする
- 例：input streamを開く
  - FileNotFoundException
- 例：input streamから読む
  - IOException

例外を呼出側に投げる

- メソッドの呼び出し時に発生した例外を呼び出し側に伝える

```
public void method() throws Exception{
....
   try{
      ....
   } catch(Exception e){
      throw e;
   }
}
```

例外を呼出側に投げる

```
public void method() throws Exception{
....
   if(条件){
      String message =”メッセージ”;
      throw new Exception(message);
   }
}
```

JDK中のソースファイルの参照

- Netbeans使用中にjdkのソースを見ることができる

- 見たいクラス名の文字列をマウスでダブルクリックして選択
- マウス右ボタン:「ナビゲート」→「ソースへ移動」