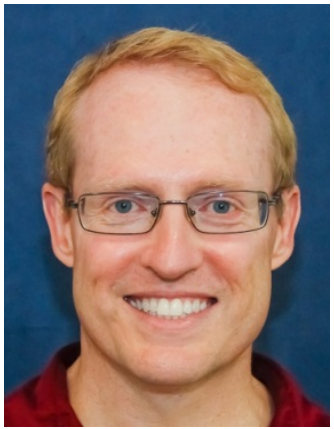
External vs. Internal Iteration in Java 8

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA



Learning Objectives in this Lesson

- Recognize the difference between external & internal iteration in Java 8

External Iteration Versus Internal Iteration

External Iteration Versus Internal Iteration

- Programmers are responsible for externally iterating through Java collections

External Iteration Versus Internal Iteration

- Programmers are responsible for externally iterating through Java collections, e.g.,

```
List<String> namesList =
  Arrays.asList("Larry",
                "Curly",
                "Moe");

for (String name : namesList)
  System.out.println(name);
```

See docs.oracle.com/javase/1.5.0/docs/guide/language/foreach.html

External Iteration Versus Internal Iteration

- Programmers are responsible for externally iterating through Java collections, e.g.,

```
List<String> namesList =
  Arrays.asList("Larry",
                "Curly",
                "Moe");

for (String name : namesList)
  System.out.println(name);
```

External Iteration Versus Internal Iteration

- Programmers are responsible for externally iterating through Java collections, e.g.,

```
List<String> namesList =
  Arrays.asList("Larry",
                "Curly",
                "Moe");

for (String name : namesList)
  System.out.println(name);
```

External Iteration Versus Internal Iteration

- Programmers are responsible for externally iterating through Java collections, e.g.,

```
List<String> namesList =
  Arrays.asList("Larry",
                "Curly",
                "Moe");

for (Iterator<String>> i =
       namesList.iterator();
     i.hasNext();)
  System.out.printLn(i.next());
```

See docs.oracle.com/javase/8/docs/api/java/util/Iterator.html

External Iteration Versus Internal Iteration

- Java 8 aggregate operations are responsible for internally iterating through Java streams

See docs.oracle.com/javase/tutorial/collections/streams/#differences

External Iteration Versus Internal Iteration

- Java 8 aggregate operations are responsible for internally iterating through Java streams, e.g.,

```
List<String> namesList =
  Arrays.asList("Larry",
                "Curly",
                "Moe");

namesList.stream().forEach
  (System.out::println);
```

See docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html#forEach

External Iteration Versus Internal Iteration

- Java 8 aggregate operations are responsible for internally iterating through Java streams, e.g.,

```
List<String> namesList =
  Arrays.asList("Larry",
                "Curly",
                "Moe");

namesList.stream().forEach
  (System.out::println);
```

External Iteration Versus Internal Iteration

- Java 8 aggregate operations are responsible for internally iterating through Java streams, e.g.,

```
List<String> namesList =
  Arrays.asList("Larry",
                "Curly",
                "Moe");

namesList.stream().forEach
  (System.out::println);
```

External Iteration Versus Internal Iteration

- Java 8 aggregate operations are responsible for internally iterating through Java streams
  - Internal iteration becomes more useful as the complexity of a stream pipeline increases

External Iteration Versus Internal Iteration

- Java 8 aggregate operations are responsible for internally iterating through Java streams
  - Internal iteration becomes more useful as the complexity of a stream pipeline increases, e.g.,


```
List<URL> urls = Stream
    .of(urlArray)
    .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
    .map(url -> { try { return new URL(url); }
                  catch (Exception ex) { ... } })
    .collect(toList());
```

External Iteration Versus Internal Iteration

- Java 8 aggregate operations are responsible for internally iterating through Java streams
  - Internal iteration becomes more useful as the complexity of a stream pipeline increases, e.g.,


```
List<URL> urls = Stream
    .of(urlArray)
    .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
    .map(url -> { try { return new URL(url); }
                  catch (Exception ex) { ... } })
    .collect(toList());
```

Checked exceptions are awkward!

See slieb.org/blog/throwable-interfaces

External Iteration Versus Internal Iteration

- Java 8 aggregate operations are responsible for internally iterating through Java streams
  - Internal iteration becomes more useful as the complexity of a stream pipeline increases, e.g.,

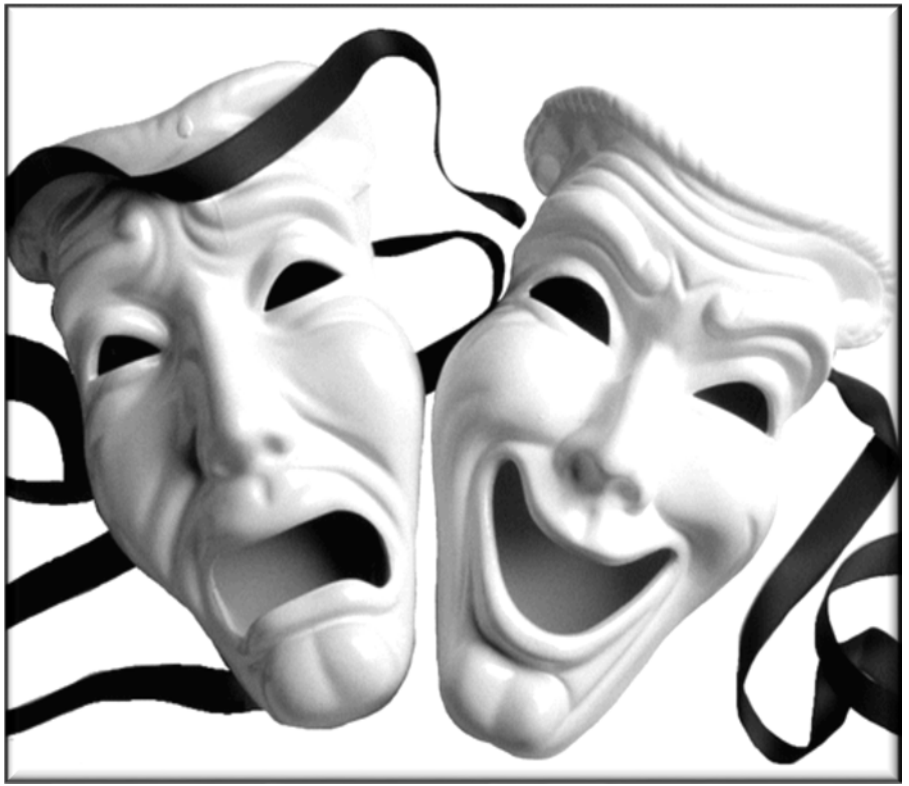

```
List<URL> urls = Stream
    .of(urlArray)
    .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
    .map(rethrowFunction(URL::new))
    .collect(toList());
```

rethrowFunction() converts checked exception into runtime exception

See stackoverflow.com/a/27661504/3312330

External Iteration Versus Internal Iteration

- Advantages of internal iterators over external iterators



See www.javabrahman.com/java-8/java-8-internal-iterators-vs-external-iterators

External Iteration Versus Internal Iteration

- Advantages of internal iterators over external iterators
  - **Improved code readability**

```
List<URL> urls = Stream
    .of(urlArray)
    .filter(s -> s.contains("cse.wustl"))
    .map(s -> s.replace("cse.wustl",
                        "dre.vanderbilt"))
    .map(rethrowFunction(URL::new))
    .collect(toList());
```

```
List<URL> urls =
  new ArrayList<URL>();
...
for (String url : urlArray)
  if (url.contains("cse.wustl"))
    urls.add(new URL(url.replace("cse.wustl",
                                 "dre.vanderbilt")));
```

External Iteration Versus Internal Iteration

- Advantages of internal iterators over external iterators
  - **Improved code readability**

```
List<URL> urls = Stream
    .of(urlArray)
    .filter(s -> s.contains("cse.wustl"))
    .map(s -> s.replace("cse.wustl",
                        "dre.vanderbilt"))
    .map(rethrowFunction(URL::new))
    .collect(toList());
```

```
List<URL> urls =
  new ArrayList<URL>();
...
for (String url : urlArray)
  if (url.contains("cse.wustl"))
    urls.add(new URL(url.replace("cse.wustl",
                                 "dre.vanderbilt")));
```

Focus on the "what" rather than the "how"

External Iteration Versus Internal Iteration

- Advantages of internal iterators over external iterators
  - **Improved code readability**

```
List<URL> urls = Stream
    .of(urlArray)
    .filter(s -> s.contains("cse.wustl"))
    .map(s -> s.replace("cse.wustl",
                        "dre.vanderbilt"))
    .map(rethrowFunction(URL::new))
    .collect(toList());
```

```
List<URL> urls =
  new ArrayList<URL>();
...
for (String url : urlArray)
  if (url.contains("cse.wustl"))
    urls.add(new URL(url.replace("cse.wustl",
                                 "dre.vanderbilt")));
```

Focus on the "what" rather than the "how"

External Iteration Versus Internal Iteration

- Advantages of internal iterators over external iterators
  - Improved code readability
  - **Transparent optimizations**

```
List<URL> urls = Stream
    .of(urlArray)
    .parallel()
    .filter(s -> s.contains("cse.wustl"))
    .map(s -> s.replace("cse.wustl",
                        "dre.vanderbilt"))
    .map(rethrowFunction(URL::new))
    .collect(toList());
```

```
List<URL> urls =
  new ArrayList<URL>();
...
for (String url : urlArray)
  if (url.contains("cse.wustl"))
    urls.add(new URL(url.replace("cse.wustl",
                                 "dre.vanderbilt")));
```

External Iteration Versus Internal Iteration

- Advantages of internal iterators over external iterators
  - Improved code readability
  - **Transparent optimizations**

```
List<URL> urls = Stream
    .of(urlArray)
    .parallel()
    .filter(s -> s.contains("cse.wustl"))
    .map(s -> s.replace("cse.wustl",
                        "dre.vanderbilt"))
    .map(rethrowFunction(URL::new))
    .collect(toList());
```

Transparently run the stream in parallel

```
List<URL> urls =
  new ArrayList<URL>();
...
for (String url : urlArray)
  if (url.contains("cse.wustl"))
    urls.add(new URL(url.replace("cse.wustl",
                                 "dre.vanderbilt")));
```

External Iteration Versus Internal Iteration

- Advantages of internal iterators over external iterators
  - Improved code readability
  - Transparent optimizations
  - **Fewer bugs**

```
List<URL> urls = Stream
    .of(urlArray)
    .filter(s -> s.contains("cse.wustl"))
    .map(s -> s.replace("cse.wustl",
                        "dre.vanderbilt"))
    .map(rethrowFunction(URL::new))
    .collect(toList());
```

```
List<URL> urls =
  new ArrayList<URL>();
...
for (String url : urlArray)
  if (url.contains("cse.wustl"))
    urls.add(new URL(url.replace("cse.wustl",
                                 "dre.vanderbilt")));
```

External Iteration Versus Internal Iteration

- Advantages of internal iterators over external iterators
  - Improved code readability
  - Transparent optimizations
  - **Fewer bugs**

```
List<URL> urls = Stream
    .of(urlArray)
    .filter(s -> s.contains("cse.wustl"))
    .map(s -> s.replace("cse.wustl",
                        "dre.vanderbilt"))
    .map(rethrowFunction(URL::new))
    .collect(toList());
```

```
List<URL> urls =
  new ArrayList<URL>();
...
for (String url : urlArray)
  if (url.contains("cse.wustl"))
    urls.add(new URL(url.replace("cse.wustl",
                                 "dre.vanderbilt")));
```

Doesn't split creation from initialization of collections

External Iteration Versus Internal Iteration

- Advantages of internal iterators over external iterators
  - Improved code readability
  - Transparent optimizations
  - **Fewer bugs**

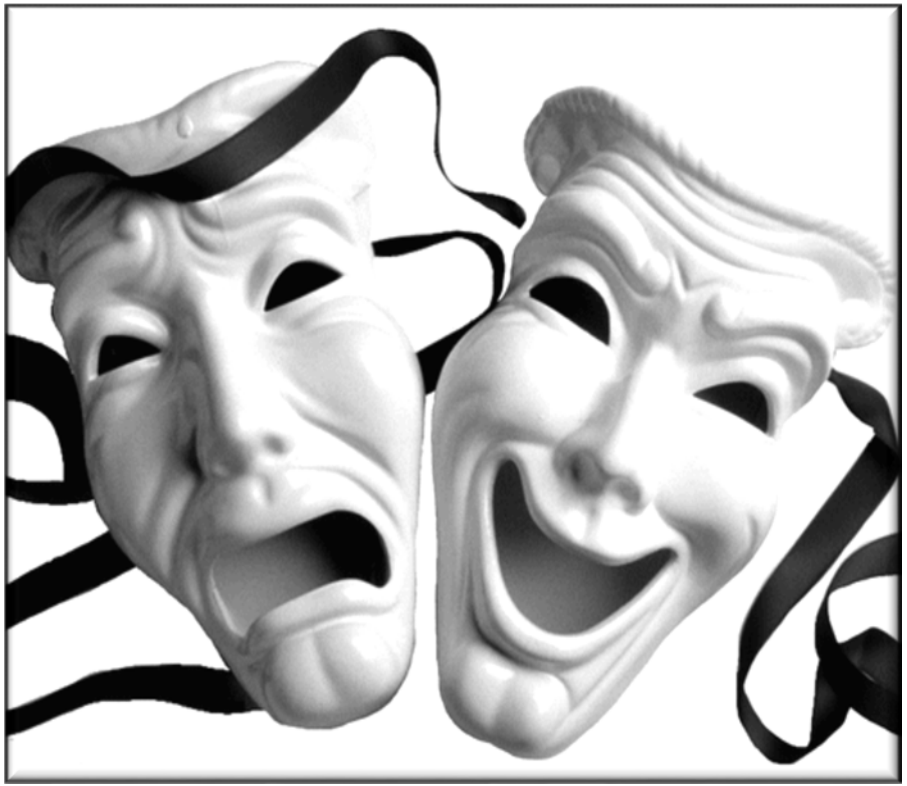
```
List<URL> urls = Stream
    .of(urlArray)
    .filter(s -> s.contains("cse.wustl"))
    .map(s -> s.replace("cse.wustl",
                        "dre.vanderbilt"))
    .map(rethrowFunction(URL::new))
    .collect(toList());
```

```
List<URL> urls =
  new ArrayList<URL>();
...
for (String url : urlArray)
  if (url.contains("cse.wustl"))
    urls.add(new URL(url.replace("cse.wustl",
                                 "dre.vanderbilt")));
```

Does split creation from initialization of collections

External Iteration Versus Internal Iteration

- Advantages of external iterators over internal iterators



See www.javabrahman.com/java-8/java-8-internal-iterators-vs-external-iterators

External Iteration Versus Internal Iteration

- Advantages of external iterators over internal iterators
  - **More control over iteration steps**

```
List<URL> urls = Stream
    .of(urlArray)
    .filter(s -> s.contains("cse.wustl"))
    .map(s -> s.replace("cse.wustl",
                        "dre.vanderbilt"))
    .map(rethrowFunction(URL::new))
    .collect(toList());
```

```
List<URL> urls =
  new ArrayList<URL>();
...
for (String url : urlArray)
  if (!url.contains("cse.wustl"))
    break;
  ...
```

External Iteration Versus Internal Iteration

- Advantages of external iterators over internal iterators
  - **More control over iteration steps**

```
List<URL> urls = Stream
        .of(urlArray)
        .filter(s -> s.contains("cse.wustl"))
        .map(s -> s.replace("cse.wustl",
                            "dre.vanderbilt"))
        .map(rethrowFunction(URL::new))
        .collect(toList());
```

```
List<URL> urls =
  new ArrayList<URL>();
...
for (String url : urlArray)
  if (!url.contains("cse.wustl"))
    break;
  ...
```

Exit a loop gracefully at an arbitrary point or handle errors more precisely

External Iteration Versus Internal Iteration

- Advantages of external iterators over internal iterators
  - More control over iteration steps
  - **Multiple active iterators**

```
List<URL> urls = Stream
    .of(urlArray)
    .filter(s -> s.contains("cse.wustl"))
    .map(s -> s.replace("cse.wustl",
                        "dre.vanderbilt"))
    .map(rethrowFunction(URL::new))
    .collect(toList());
```

```
for (;;) {
  Iterator<URL>> iter1 = urls.iterator();
  Iterator<URL>> iter2 = urls.iterator();

  if (iter1.hasNext()) { URL url = iter1.next(); ... }
  if (iter2.hasNext()) { URL url = iter2.next(); ... }
  ...
```

Many iterators can be active over the same object

End of External Iteration vs. Internal Iteration