External vs. Internal Iterators in Java 8: Evaluating Pros & Cons

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 Part of the Lesson

- Recognize the difference between external & internal iterators in Java 8
- Know how to evaluate the pros & cons of external vs. internal iterators

Advantages of Internal Iterators Over External Iterators

Advantages of Internal Iterators over External Iterators

- Advantages of internal iterators over external iterators

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

Advantages of Internal Iterators over External Iterators

- 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")));
```

Advantages of Internal Iterators over External Iterators

- 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());
```

Focus on the "what" rather than the "how," e.g., no control flow operations.

```
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")));
```

Advantages of Internal Iterators over External Iterators

- 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")));
```

More focus on the "how," e.g., Java control flow operations.

Advantages of Internal Iterators over External Iterators

- 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")));
```

Advantages of Internal Iterators over External Iterators

- 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")));
```

Advantages of Internal Iterators over External Iterators

- 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")));
```

Always runs sequentially (Accumulator "anti-pattern")

See www.ibm.com/developerworks/library/j-java-streams-2-brian-goetz

Advantages of Internal Iterators over External Iterators

- 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")));
```

Advantages of Internal Iterators over External Iterators

- 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());
```

Doesn't split creation from initialization of collections

```
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")));
```

Advantages of Internal Iterators over External Iterators

- 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

Advantages of External Iterators Over Internal Iterators

Advantages of External Iterators over Internal Iterators

- Advantages of external iterators over internal iterators

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

Advantages of External Iterators over Internal Iterators

- 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;
  ...
```

Advantages of External Iterators over Internal Iterators

- 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;
  ...
```

Can't exit the stream without throwing an exception..

Advantages of External Iterators over Internal Iterators

- 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.

Advantages of External Iterators over Internal Iterators

- 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

Advantages of External Iterators over Internal Iterators

- 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());
```

Only one (internal) iterator for a stream

```
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(); ... }
  ...
```

End of External Iterators vs. Internal Iterators: Evaluating Pros & Cons