Contrasting Java Streams with Java Collections

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

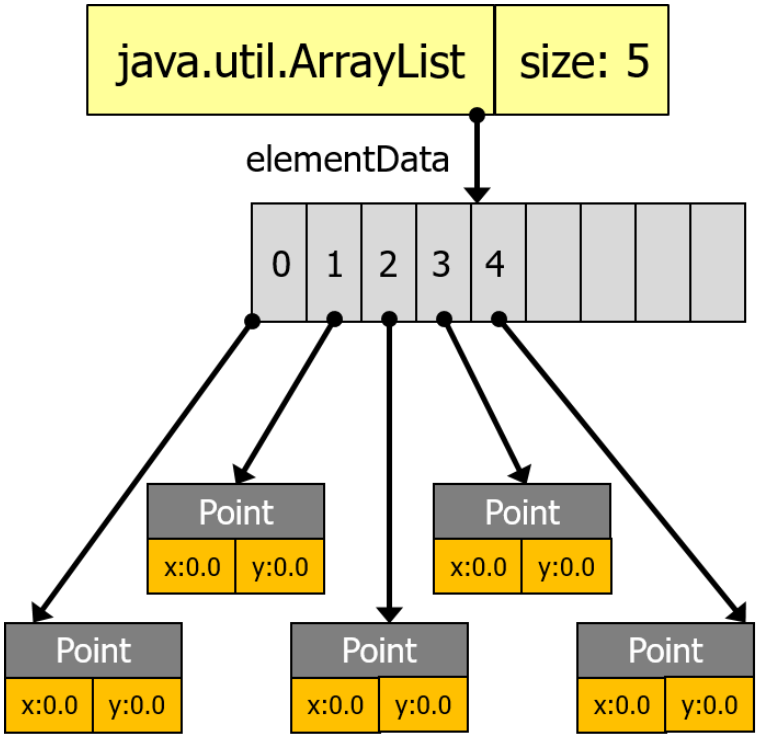
Professor of Computer Science

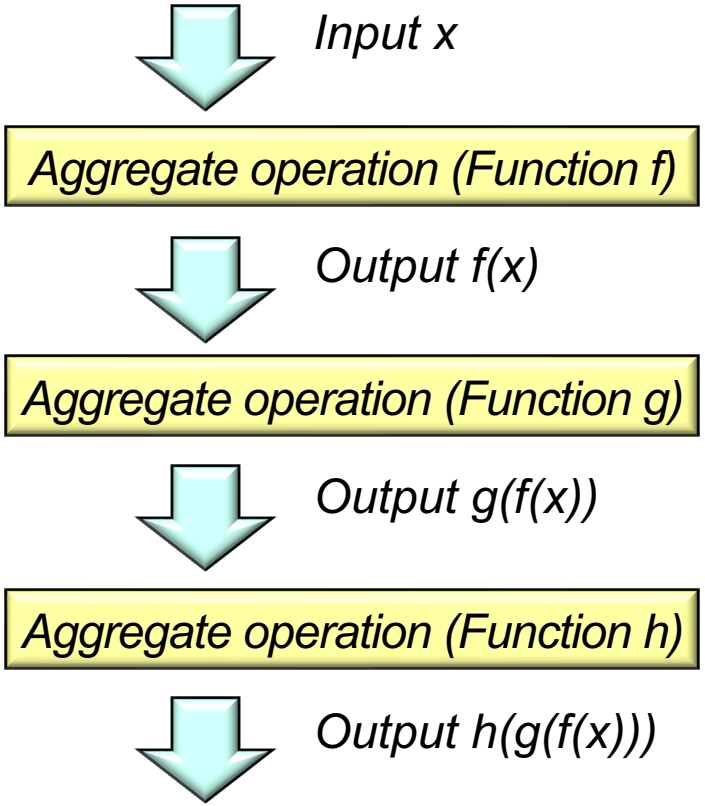
Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand how Java collections contrast with Java streams





Learning Objectives in this Part of the Lesson

- Understand how Java collections contrast with Java streams
  - Know how to program with Java collections & streams

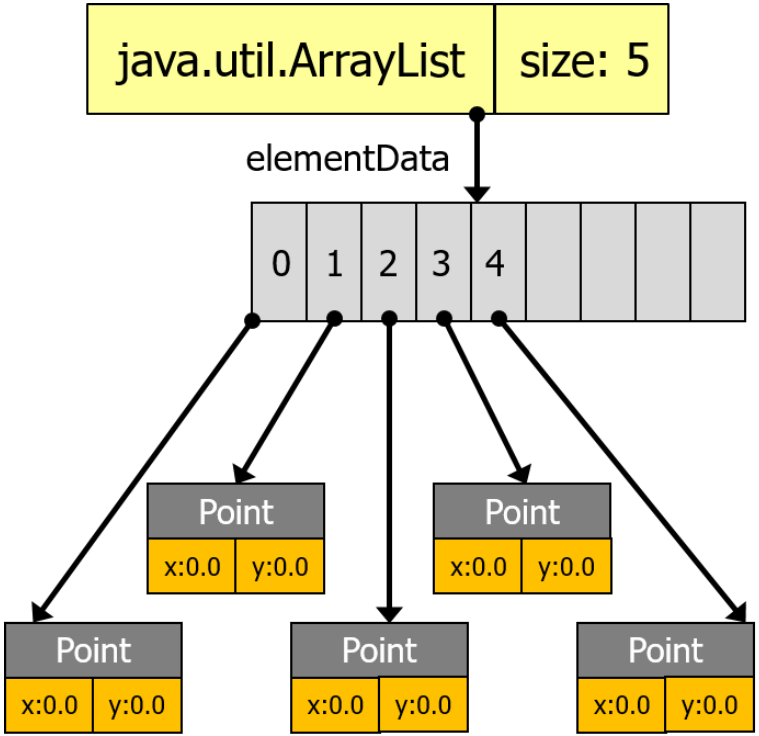
```
List<String> urls = new ArrayList<>(Arrays.asList(urlArray));
for (int i = 0; i < urls.size(); ++i) {
   if (!urls.get(i).contains("cse.wustl"))
   { urls.remove(i); continue; }
   urls.set(i,
            urls.get(i).replace("cse.wustl","dre.vanderbilt"));
}
```

```
List<String> urls = Arrays.asList(urlArray)
   .stream()
   .filter(s -> s.contains("cse.wustl"))
   .map(s -> s.replace("cse.wustl", "dre.vanderbilt"))
   .collect(toList());
```

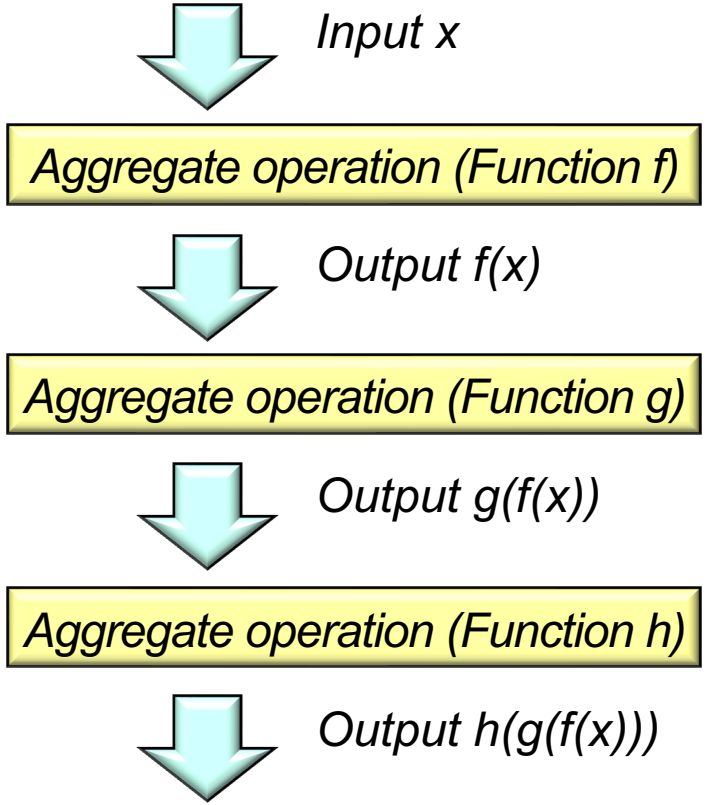
Contrasting Java Collections & Java Streams

Contrasting Java Collections & Java Streams

- Java collections are different from Java streams!



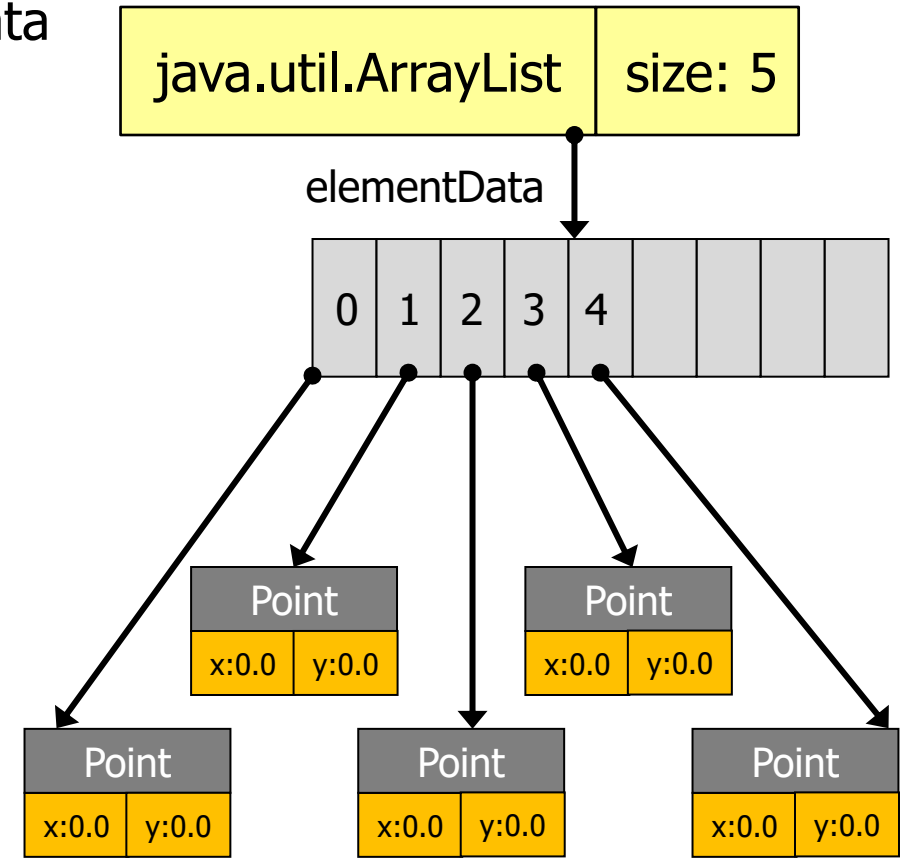
≠



See www.oracle.com/technetwork/articles/java/ma14-java-se-8-streams-2177646.html

Contrasting Java Collections & Java Streams

- A Java collection is an in-memory data structure that can store, retrieve, & manipulate groups of elements



See docs.oracle.com/javase/tutorial/collections/intro

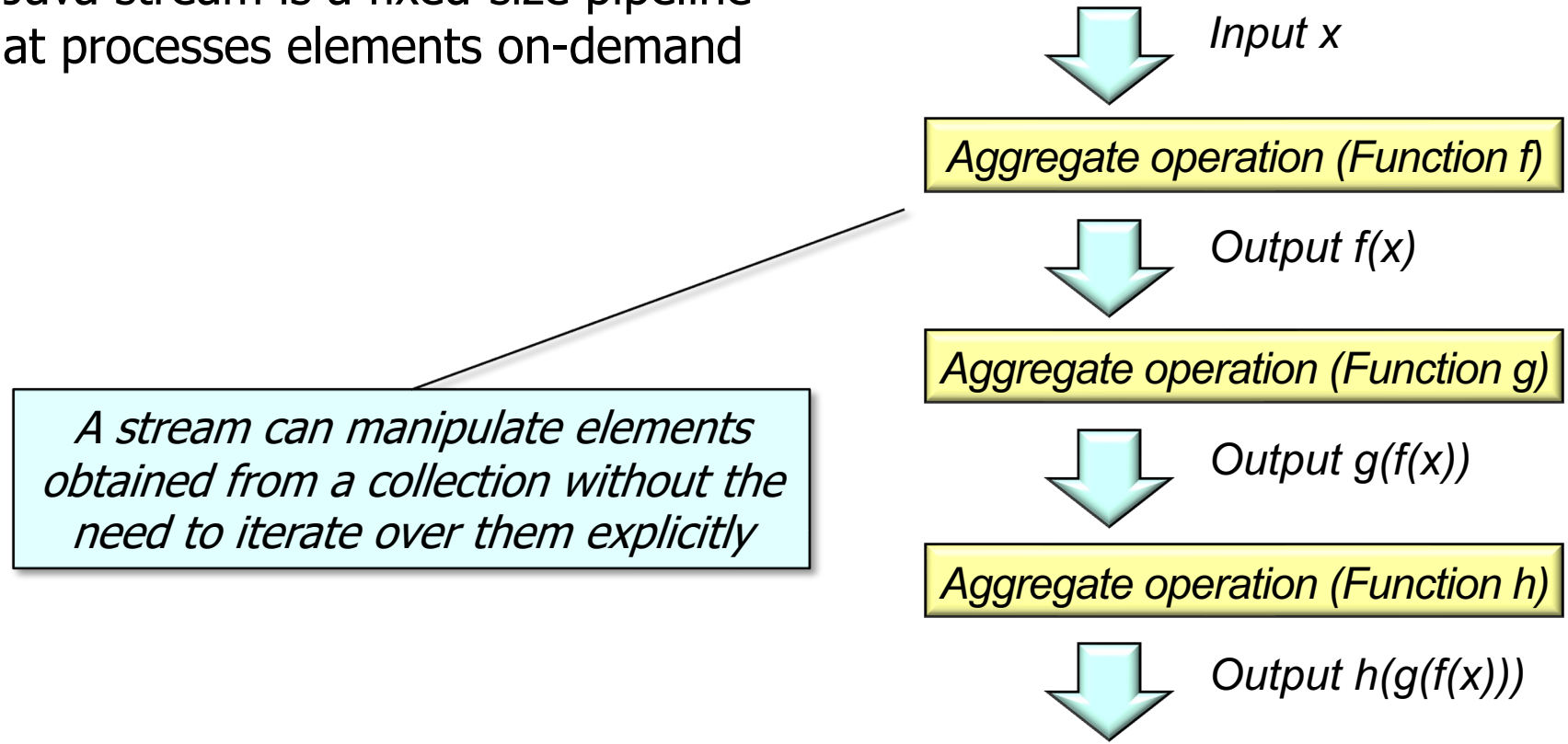
Contrasting Java Collections & Java Streams

- A Java collection is an in-memory data structure that can store, retrieve, & manipulate groups of elements
  - It is somewhat analogous to the contents on a DVD

e.g., its content can be read from & written to (sometimes)

Contrasting Java Collections & Java Streams

- A Java stream is a fixed-size pipeline that processes elements on-demand



See tutorials.jenkov.com/java-collections/streams.html

Contrasting Java Collections & Java Streams

- A Java stream is a fixed-size pipeline that processes elements on-demand
  - It is somewhat analogous to a flow of elements in a video stream

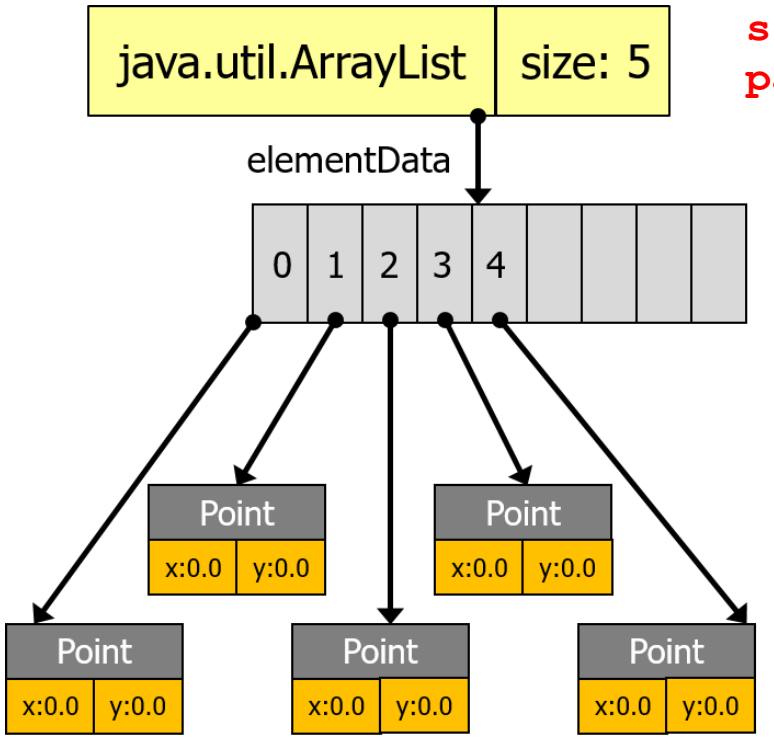
e.g.., its content is processed as it flows through the stream

Contrasting Java Collections & Java Streams

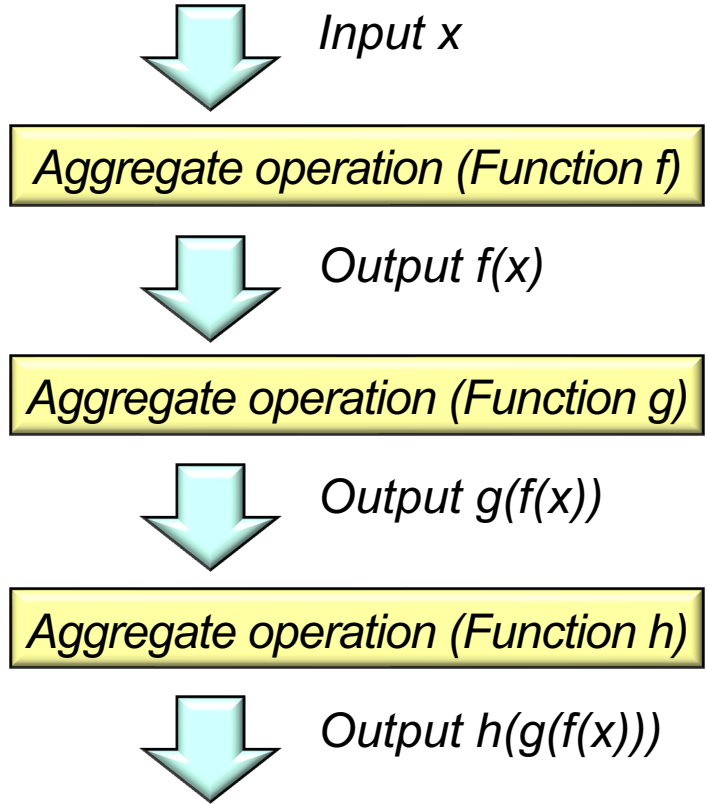
- These analogies are not perfect!
  - e.g., collections generally are persistent & streams generally aren't infinite

Contrasting Java Collections & Java Streams

- Various factory methods can convert collections to streams



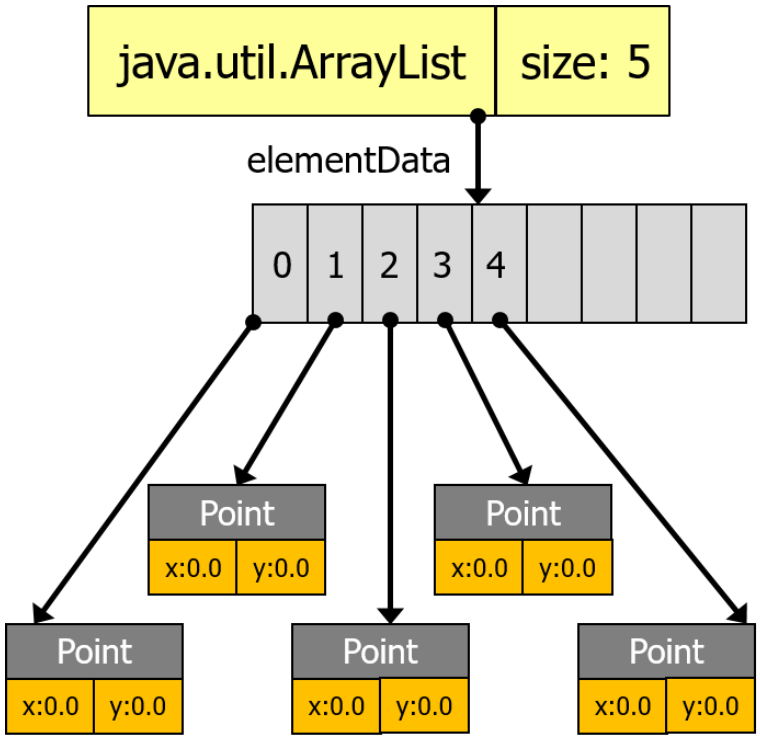
`stream()`
`parallelStream()`



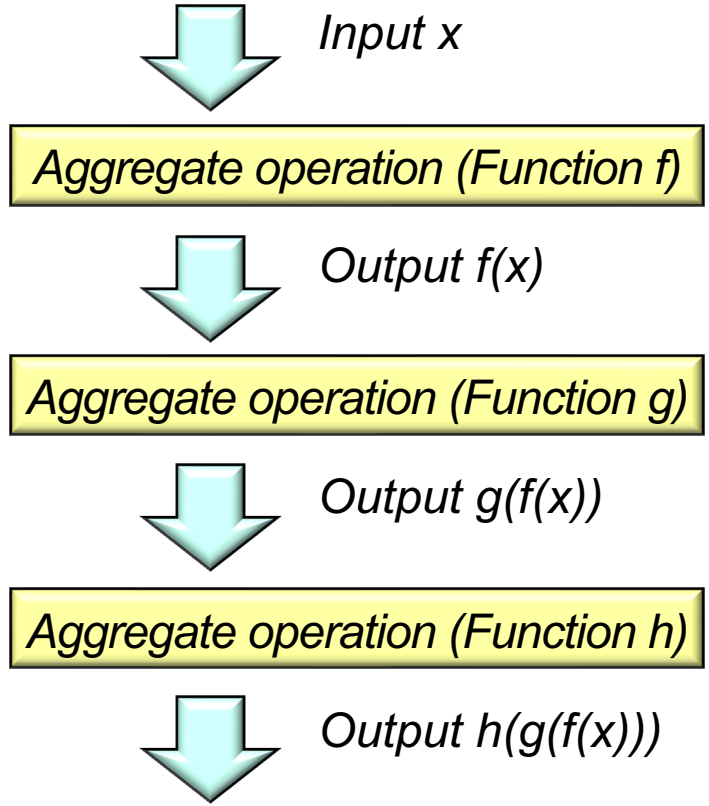
See upcoming lessons on "*Stream Creation Operations*"

Contrasting Java Collections & Java Streams

- Other terminal operations can convert streams to collections



`collect()`



See upcoming lessons on "*Stream Terminal Operations*"

Examples of Java Collections & Java Streams

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java collection

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

List<String> urls = new ArrayList<>(Arrays.asList(urlArray));


for (int i = 0; i < urls.size(); ++i) {
  if (!urls.get(i).contains("cse.wustl"))
  { urls.remove(i); continue; }
  urls.set(i,
           urls.get(i).replace("cse.wustl","dre.vanderbilt"));
}
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex41

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java collection

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

List<String> urls = new ArrayList<>(Arrays.asList(urlArray));

for (int i = 0; i < urls.size(); ++i) {
  if (!urls.get(i).contains("cse.wustl"))
  { urls.remove(i); continue; }
  urls.set(i,
           urls.get(i).replace("cse.wustl","dre.vanderbilt"));
}
```

This example demonstrates external iteration

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

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java collection

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

List<String> urls = new ArrayList<>(Arrays.asList(urlArray));


for (int i = 0; i < urls.size(); ++i) {
  if (!urls.get(i).contains("cse.wustl"))
  { urls.remove(i); continue; }
  urls.set(i,
           urls.get(i).replace("cse.wustl","dre.vanderbilt"));
}
```

Create a list from an array

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java collection

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

List<String> urls = new ArrayList<>(Arrays.asList(urlArray));


for (int i = 0; i < urls.size(); ++i) {
  if (!urls.get(i).contains("cse.wustl"))
  { urls.remove(i); continue; }
  urls.set(i,
           urls.get(i).replace("cse.wustl","dre.vanderbilt"));
}
```

Explicitly iterate through a list

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java collection

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

List<String> urls = new ArrayList<>(Arrays.asList(urlArray));

for (int i = 0; i < urls.size(); ++i) {
  if (!urls.get(i).contains("cse.wustl"))
  { urls.remove(i); continue; }
  urls.set(i,
           urls.get(i).replace("cse.wustl","dre.vanderbilt"));
}
```

External iteration enables fine-grained control of loop behavior

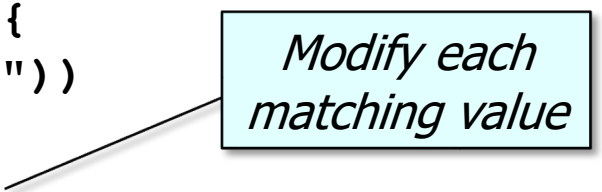
Examples of Java Collections & Java Streams

- A simple example of manipulating a Java collection

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

List<String> urls = new ArrayList<>(Arrays.asList(urlArray));


for (int i = 0; i < urls.size(); ++i) {
  if (!urls.get(i).contains("cse.wustl"))
  { urls.remove(i); continue; }
  urls.set(i,
           urls.get(i).replace("cse.wustl","dre.vanderbilt"));
}
```



Examples of Java Collections & Java Streams

- A simple example of manipulating a Java collection

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

List<String> urls = new ArrayList<>(Arrays.asList(urlArray));
```

However, this code is tedious & error-prone to write, read, & optimize

```
for (int i = 0; i < urls.size(); ++i) {
  if (!urls.get(i).contains("cse.wustl"))
  { urls.remove(i); continue; }
  urls.set(i,
           urls.get(i).replace("cse.wustl","dre.vanderbilt"));
}
```

See howtodoinjava.com/java8/internal-vs-external-iteration

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

 List<String> urls = Arrays.asList(urlArray)
   .stream()
   .filter(s -> s.contains("cse.wustl"))
   .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
   .toList();
```

This example shows the "fluent interface" programming style, internal iteration, chaining of transformations

See en.wikipedia.org/wiki/Fluent_interface

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

List<String> urls = Arrays.asList(urlArray)
  .stream()
  .filter(s -> s.contains("cse.wustl"))
  .map(s ->
       s.replace("cse.wustl", "dre.vanderbilt"))
  .toList();
```

Implicitly iterate through a pipeline of elements from a collection source, filter/transform each value, & create a collection result

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

 List<String> urls = Arrays.asList(urlArray)
   .stream()
   .filter(s -> s.contains("cse.wustl"))
   .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
   .toList();
```

Implicitly iterate through a pipeline of elements from a collection source, filter/transform each value, & create a collection result

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

List<String> urls = Arrays.asList(urlArray)
   .stream()
   .filter(s -> s.contains("cse.wustl"))
   .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
   . toList();
```

Implicitly iterate through a pipeline of elements from a collection source, filter/transform each value, & create a collection result

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

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

Java streams simplifies chaining of transformations

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

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

rethrowFunction() converts checked exception into runtime exception

See stackoverflow.com/a/27661504/3312330

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

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

Parallelizing a Java stream is often easy!!

See docs.oracle.com/javase/tutorial/collections/streams/parallelism.html

Examples of Java Collections & Java Streams

- A simple example of manipulating a Java stream

```
String[] urlArray = {
  "http://www.cse.wustl.edu/~schmidt/gifs/ka.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/robot.png",
  "http://www.cse.wustl.edu/~schmidt/gifs/kitten.png"};

 List<String> urls = Arrays.asList(urlArray)
   .stream()
   .filter(s -> s.contains("cse.wustl"))
   .map(s ->
        s.replace("cse.wustl", "dre.vanderbilt"))
   .toList();
```

Like iterators, elements in a stream can only be visited once during its lifetime

End of Contrasting Java Streams with Java Collections