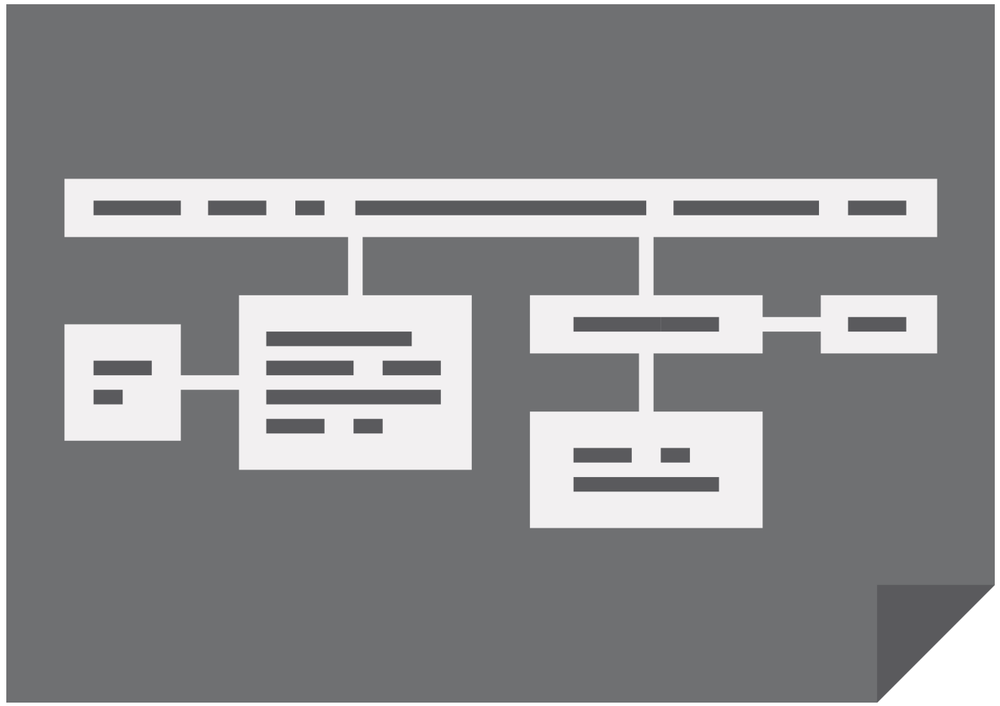
Iterator Pattern

Bryan Hansen

twitter: bh5k | http://www.linkedin.com/in/hansenbryan

Concepts

- Traverse a container
- Doesn’t expose underlying structure
- Decouples algorithms
- Sequential
- Examples:
  - java.util.Iterator
  - java.util.Enumeration



Design

Interface based

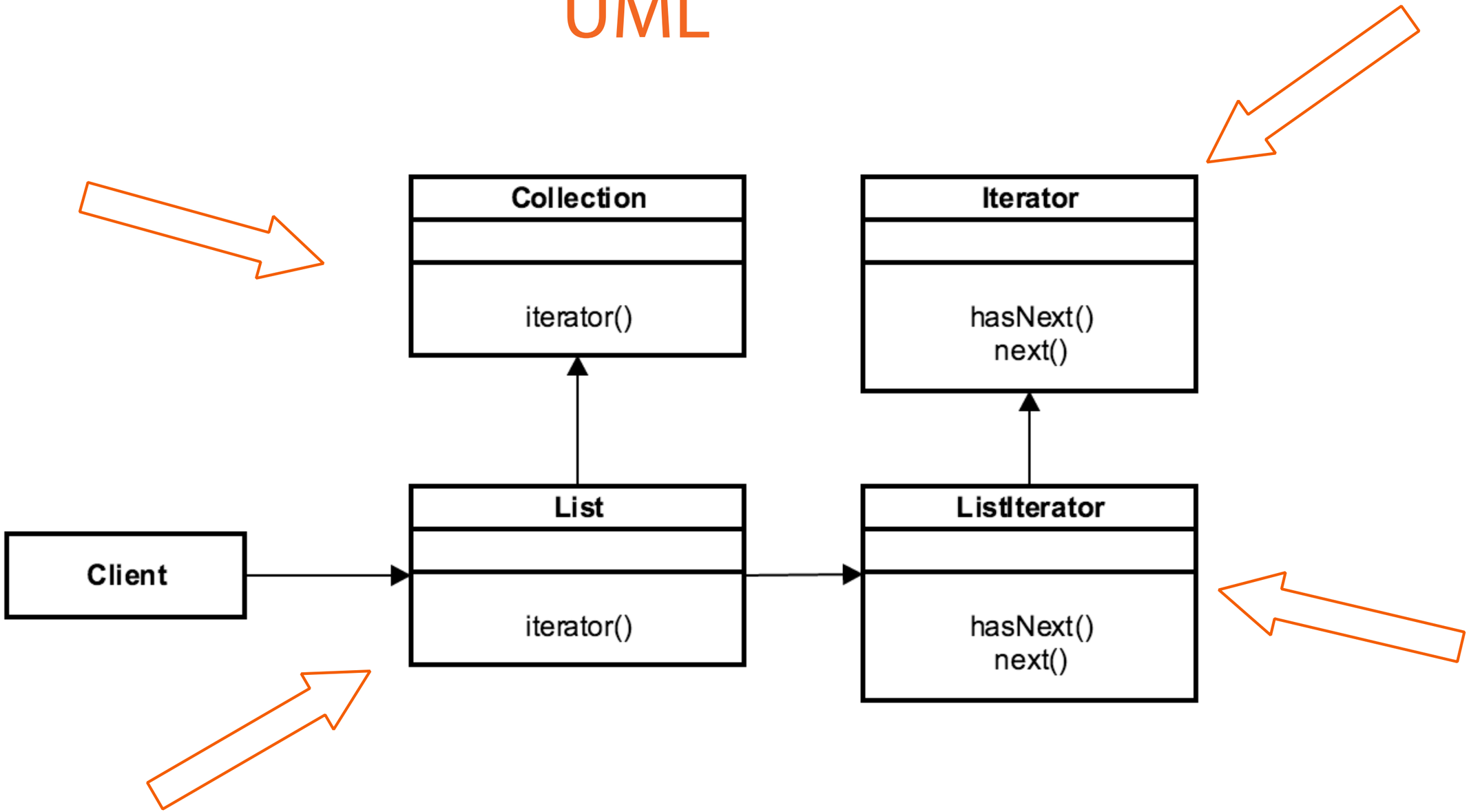
Factory Method based

Independent, but fail fast

Enumerators are fail safe

Iterator, ConcreteIterator

UML


Collection
iterator()
Iterator
hasNext()
next()
List
iterator()
ListIterator
hasNext()
next()
Client

Everyday Example - List

```
List<String> names = new ArrayList<>();

names.add("Bryan");
names.add("Aaron");
names.add("Jason");

Iterator<String> namesItr = names.iterator();

while(namesItr.hasNext()) {
   String name = namesItr.next();
   System.out.println(name);
   namesItr.remove();
}
```

Exercise Iterator

Repository

Iterator

Pitfalls

- Access to Index
- Directional
- Speed / Efficiency

Contrast

Iterator

- Interface based
- Algorithm is removed
- No index
- Concurrent modification

For loop

- Traversal in client
- Exposes an index
- Doesn’t change underlying object
- foreach syntax
- Typically slower

Iterator Summary

- Efficient way to traverse
- Hides algorithm
- Simplify client
- Foreach