Iterator

Iterator

- Iterator defines an interface for sequencing through the objects in a collection.
  - This interface is independent of the details of the kind of collection and its implementation.
- This pattern is applicable to any language

External Iterators

- In languages without closures, we are forced to use external iterators, *e.g.*, in Java:
  - aCollection.iterator() answers an iterator.
  - the programmer must explicitly manipulate the iterator with a loop using hasNext() and next()

Java test

- Given a collection of integers, answer a similar collection containing their squares:

your answer here ...

Internal Iterators

- Languages with closures provide a better way of writing iterators
- Internal Iterators encapsulate the loop itself, and the next and hasNext operations in a single method
- Examples: do:, collect:, inject:into:
  - look at the enumerating protocol in Collection

doing: Iterators for effect

For every (or most) elements in the collection, do some action

do: do:separatedBy: do:without:

- for keyedCollections

  associationsDo: keysDo: valuesDo:

- for SequenceableCollections

  withIndexDo: reverseDo: allButFirstDo:

mapping: create a new collection

- Create a new collection of the same kind as the old one, with elements in one-to-one correspondence
- For every element in the collection, create a new element for the result.

  collect: collect:thenDo: collect:thenSelect:

- for SequenceableCollections

  collect:from:to: withIndexcollect:

selecting: filtering a collection

- Create a new collection of the same kind as the old one, with a subset of its elements
- For every element in the collection, apply a filter.
- Examples:

```
select:              reject:
select:thenDo:       reject:thenDo:
```

partial do

- It's OK to return from the block that is the argument of a do:

```
coll do: [ :each | each matches: pattern ifTrue: [^ each]].
^ default
```

  - but consider using one of the "electing" iterators first!

```
coll detect: [ :each | each matches: pattern]
      ifNone: [default]
```

electing: picking an element

Choose a particular element that matches some criterion

- Criterion might be fixed:
  - max: min:
- or programmable:
  - detect: detect:ifNone:

Summarizing: answering a single value

- Answer a single value that tells the client something about the collection
  - allSatisfy: anySatisfy:
    detectMin: detectMax: detectSum:
  - sum inject: into: