Data

- id : int
- content : String

+ getId() : int
+ getContent() : String
+ hashCode() : int
+ equals(Object obj) : boolean

«interface»
IData

+ getId() : int
+ getContent() : String

DataProcessingApp

+ main(String[] args)

DataGenerator

- filter : IFilter

+ DataGenerator(int low, int high, IFilter filter)
+ iterator() : Iterator

«interface»
IFilter

+ canFilter(IData d) : boolean
+ filter(IData d)
+ process(int low, int high)
+ createData(int i): data
+ createIterator(): Iterator

PrimeFilter

- data : List

+ canFilter(IData d) : boolean
+ filter(IData d)
+ process(int low, int high)
+ createData(int i): data
+ createIterator(): Iterator

«interface»
Iterable

+ iterator() : Iterator

«interface»
Iterator

+ hasNext() : boolean
+ next() : E
+ remove()

PrimeFilterIterator

- list : List
- position : int

+ PrimeFilterIterator(List data)
+ hasNext() : boolean
+ next() : Object
+ remove()

The DataProcessingApp iterates over each IData object in the DataGenerator to display each object's ID and content. In order for this to be possible, the DataGenerator's constructor passes the low and high ints to the filter by calling process on the given filter. The given filter in this case is the PrimeFilter which creates the appropriate Data objects after receiving the two ints and places the Data instances into its List. The List is then passed to the PrimeFilterIterator to create an Iterator for the IData objects. Additional filters would implement the IFilter interface and also implement their own Iterators, relieving the control of execution to the framework.