

Iterator/ Comparables

Iterators

- An iterator is an object that allows you to transverse through an collection of items one at a time.
- In data structures that don't allow accessing elements using index, the only way to iterate through the elements are by using iterators.
- A class that is iterable in the JCF implements the Iterable interface (<https://docs.oracle.com/javase/7/docs/api/java/lang/Iterable.html>). The interface only has one method that returns an iterator object.
- The iterator class is implements the Iterator interface(<https://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html>) which contains 3 methods:
 - hasNext() -> returns the next element
 - next() -> returns true if there is a next element
 - remove() -> removes the last seen elements, typically not implemented
 - These methods should run O(1)

```
import java.util.Iterator;

public class IterableObject<T> implements Iterable<T>{
    public Iterator<T> iterator(){
        return new ObjectIterator<T>(front);
    }
}
```

```
class ObjectIterator<T> implements Iterator<T>{

    private T current;

    public ObjectIterator(T c){
        current = c;
    }

    public T next(){
        T answer = current.getData();
        current = current.getNext();
        return answer;
    }

    public boolean hasNext(){
        return current!=null;
    }
}
```

```

public void remove(){
    throw new UnsupportedOperationException();
}
}

```

ForEach Loop

- Java allows for-each loops

```

for([object] : [structure])
    [action];

```

- Example of for each:

```

for(Integer x : s){
    System.out.println(x);
}

```

- This translates to:

```

Iterator<Integer> iter = S.iterator();
while(iter.hasNext())
    System.out.println(iter.next());

```

- For each *object* in the *structure* perform this *action*
- In order for this to work, the structure must be Iterable
- However, you cannot remove elements during a for-each loop

Comparable/Comparator

- Java provides two interfaces to sort objects using data members:
 - Comparable (<https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html>)
 - Comparator (<https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html>)
- Of the two, we will be using Comparable since it is much easier to use and understand.

Comparable

- A comparable object is capable of comparing itself to another object. The objects class itself implements the Comparable interface to compare its instances
- The comparable interface only has one method, *compareTo(T y)*
- The compareTo method compares the object(x) to another object(y) being passed in by:
 - If $x < y$ returns an integer < 0
 - If $x > y$ returns an integer > 0
 - If $x == y$ returns 0