CSE12 - Lecture 22 - C00

Tuesday, November 15, 2022 11:00 AM

Exam 2 → Friday → run-time → hash tables
PA8 released today → discussion at 4pm
PA5 Late/Resubmit → due Tuesday
PA7 hard deadline today

Iterators

What is an iterator used for in Java?

loop through / visit, in some order, every element of a collection
↳ use in a for-each loop

What is the interface needed for creating an iterator?

$\boxed{\text{Iterable<E>}}$ Iterable<Integer>

What method(s) do we need to implement for that interface?

Iterator<E> iterator() {}

Iterator<Integer> iterator()

What class do we need to create to hold the iterators state?

Iterator

Where should that class be created?

private inner class

inside our collection or data structure

What interface does it need to implement?

Iterator<E>

Iterator<Integer>

What method(s) do we need to implement for that interface?

E next() Integer next()

boolean hasNext()

What is the process to iterate over an object? (next method)

① save the current value into temp variable

② move to the next item (update state)

③ return the temp value

```
class MyClass<E> implements Iterable<E> {
    class MyIterator<E> implements Iterator<E> {
        // state
        public MyIterator( ——— ) {
            // save initial state
        public MyIterator( ——— ) {
            // save initial state
        }

        public E next() {
            return null;
        }

        public boolean hasNext() {
            return false;
        }
    }
    public Iterator<E> iterator() {
        return new MyIterator<E>();
    }
}
```

How could we make our linked list work in an enhanced for loop? What changes would we need to make to the LList class?

```
LList<Integer> list = new LList<Integer>();

//code to add data to list

for (Integer i: list) {
  System.out.println(i);
}
```

```
Integer i;
while (list.hasNext()) {
    i = list.next();
    s.o.p(i);
}
```

should be an iterator

```
public class LList<E> implements Iterable<E> {
  Node front;
  int size;
  boolean changed = false;

  LList() { //... }
  public void prepend(E value) { //... }    changed = true;
  public E get(int index) { //... }
  public int size() { //... }
  public Iterator<E> iterator() {
      return new LLIterator<E>();
  }
  class LLIterator<E> implements Iterator<E> {
      // state
      Node<E> current;
      public LLIterator() {
          current = front.next;   //skip the dummy node
          changed = false;
      }
      public boolean hasNext() {
          return current != null;
      }
      public E next() {
                              if (changed)
                                  // throw an exception
       ① E temp = current.value;
       ② current = current.next;
       ③ return temp;
      }
  }

}
```

```
class Node<E> {
  E value;
  Node<E> next;
  public Node(E value, Node<E> next) {
    this.value = value;
    this.next = next;
  }
}
```