```java

import components.queue.QueueSecondary;

/**
 * {@code Queue} represented as a {@code Sequence} of entries, with
 * implementations of primary methods.
 *
 * @param <T> type of {@code Queue} entries
 * @correspondence this = $this.entries
 */
public abstract class Queue3<T> extends QueueSecondary<T> {

    /*
     * Private members --------------------------------------------------------
     */

    /**
     * Entries included in {@code this}.
     */
    private Sequence<T> entries;

    /**
     * Creator of initial representation.
     */
    private void createNewRep() {
        this.entries = new Sequence1L<T>();
    }

    /*
     * Constructors -----------------------------------------------------------
     */

    /**
     * No-argument constructor.
     */
    public Queue3() {
        this.createNewRep();
    }

    /*
     * Standard methods removed to reduce clutter...
     */

    /*
     * Kernel methods ---------------------------------------------------------
     */

    @Override
    public final void enqueue(T x) {
        assert x != null : "Violation of: x is not null";

        this.enqueue(x);

    }

    @Override
    public final T dequeue() {
        assert this.length() > 0 : "Violation of: this /= <>";

```

```
62          T removed = this.dequeue();
63
64          return removed;
65      }
66
67      @Override
68      public final int length() {
69
70          // This line added just to make the component compilable.
71          return this.length();
72      }
73
74      /**
75       * Reports the front of {@code this}.
76       *
77       * @return the front entry of {@code this}
78       * @aliases reference returned by {@code front}
79       * @requires this /= <>
80       * @ensures <front> is prefix of this
81       */
82      @Override
83      public T front() {
84          assert this.length() > 0 : "Violation of: this /= <>";
85
86          T front = this.dequeue();
87          T temp = front;
88
89          for (int idx = 0; idx <= this.length(); idx++) {
90              this.enqueue(temp);
91              temp = this.dequeue();
92          }
93
94          return front;
95      }
96
97      /*
98       * Iterator removed to reduce clutter...
99       */
100
101 }
```