Queues

More with Comparable

Priority Queues
Queues
Queueing Theory

Agner Krarup Erlang
Queue Interface

- null is returned from `getFront()` and `dequeue()` when queue is empty
- return false from `offer` when cannot add to queue.

```java
public interface QueueInterface<E> {
    int size();
    boolean isEmpty();
    E getFront();  // peek
    boolean enqueue(E e);
    E dequeue();
    void clear();
}
```
## Example

<table>
<thead>
<tr>
<th>Operation</th>
<th>output</th>
<th>Queue Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>enqueue(5)</td>
<td>TRUE</td>
<td>{5}</td>
</tr>
<tr>
<td>enqueue(3)</td>
<td>TRUE</td>
<td>{5, 3}</td>
</tr>
<tr>
<td>dequeue()</td>
<td>5</td>
<td>{3}</td>
</tr>
<tr>
<td>enqueue(7)</td>
<td>TRUE</td>
<td>{3, 7}</td>
</tr>
<tr>
<td>dequeue()</td>
<td>3</td>
<td>{7}</td>
</tr>
<tr>
<td>getFront()</td>
<td>7</td>
<td>{7}</td>
</tr>
<tr>
<td>dequeue()</td>
<td>7</td>
<td>{}</td>
</tr>
<tr>
<td>dequeue()</td>
<td>null</td>
<td>{}</td>
</tr>
</tbody>
</table>
Array-based Queue

- An array of size $n$ in a circular fashion
  - `frontLoc`: index of the front element
    - where objects are read
  - `count`: number of stored elements
  - `rearLoc`: index of rear element
    - where objects are added

![Diagram of normal and wrapped-around configurations of a queue with an array of size $n$.](image-url)
Circular Array and Queue

- When the queue has fewer than $n$ elements, location...
Performance and Limitations for array-based Queue

- **Performance**
  - let $n$ be the number of objects in the queue
  - The space used is $O(n)$
  - Each operation runs in time $O(1)$

- **Limitations**
  - Max size is limited and can not be changed
  - Adding to a full queue returns false (offer method)
public class ArrayQueue<Q> implements QueueInterface<Q> {
    /** the default capacity for the backing array */
    private static final int CAPACITY = 40;
    /** The array in which the queue data is stored */
    private Q[] backingArray;
    /** The number of items in the queue */
    private int count;
    /** The array location of the end of the queue (ie the
     * location of the item shown by the peek command) */
    private int frontLoc;
    public ArrayQueue(int qSize) {
        count = 0;
        frontLoc = 0;
        backingArray = (Q[]) new Object[qSize];
    }
}
Java Documentation
Queue offer Method (enqueue)

boolean offer(E e)
Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions. When using a capacity-restricted queue, this method is generally preferable to add(E), which can fail to insert an element only by throwing an exception.

Parameters:
e - the element to add

Returns:
true if the element was added to this queue, else false
public class ArrayQueue<Q> implements QueueInterface<Q> {
    /** the default capacity for the backing array */
    private static final int CAPACITY = 40;
    /** The array in which the queue data is stored */
    private Q[] backingArray;
    /** The number of items in the queue */
    private int count;
    /** The array location of the end of the queue (ie the * location of the item shown by the peek command) */
    private int frontLoc;

    boolean enqueue(E e);  // add item to queue
    E dequeue();  // remove item from queue
Comparable Rabbit

```java
public class Rabbit implements Comparable<Rabbit> {
    private final int iD;
    private final String nickname;
    public Rabbit(int id, String nn) {
        this.iD = id;
        this.nickname = nn == null ? makeName() : nn;
    }

    // implement Comparable interface so that rabbits
    // are sorted based on their iD.
```
Priority Queue

• Rather than FiFo, remove items according to their priority
  • Implement:
    • same methods as queue(?)
  • Others needed?

```java
public class PriorityQueue<B extends Comparable<B>>
    extends ArrayList<B>
    implements QueueInterface<B>

public class PriorityQueueSAL<P extends Comparable<P>>
    extends SALextending<P>
    implements QueueInterface<P>
```
PriorityQueue

• Implementation a trivial extension on SAL!!!

• Small difference
  • Usually PQ are on K,V pairs where
    • K — the priority
    • V — the item in the queue

• Q: Does K,V pair matter?