Andreea  Kelsey  Sam
Liam  Kristen  Gabby
Bei  Tolshoi
Samah  Wednesday, April 24, 12:15
Kendale 307
Cade  Angela
Jessie  Marc
Claudia

End of Year
Celebration Lunch!!
Wednesday, April 24, 12:15
Kendale 307

Stack vs. Queue
- Stack is Last In First Out (LIFO)

Queue is First In First Out (FIFO)

Queue Examples
- Deli counter
- Checkout line
- Ticket window
- Calls on hold
- Cards in a draw/discard pile
- Printer queue
- Processes ready for CPU
- Data to be sent over network

Monday, April 22, 13
Queue

- Data structure that removes items in the same order they are added

```java
public class CallQueue {
    public void enqueue (Call item);  // Add a caller to end
    public Call front();    // Get caller waiting the longest
    public Call dequeue();  // Remove caller waiting the longest
    public boolean isEmpty();  // Are any callers waiting?
    public int size();    // How many callers are waiting?
}
```

Queue Generic

- Data structure that removes items in the same order they are added

```java
public interface Queue<T> {
    public void enqueue (T item);  // Add an item to end
    public T front();    // Get item waiting the longest
    public T dequeue();  // Remove item waiting the longest
    public boolean isEmpty();  // Are any items waiting?
    public int size();    // How many items are waiting?
}
```

Managing Queue Data

- We need to know:
  - What is at the front of the queue
  - Maintain elements in the order added
  - Where the next thing added to the queue should go
  - How should we organize the data?
    - Array?
    - ArrayList?
    - Linked list?

Monday, April 22, 13
Queue Implemented with a Linked List

head = null
tail = null

enqueue (1);

Queue Implemented with a Linked List

head ——— tail

enqueue (1);

enqueue (2);

Queue Implemented with a Linked List

head ——— tail

enqueue (1);
enqueue (2);

Monday, April 22, 13
Queue Implemented with a Linked List

```
enqueue (1);
enqueue (2);
enqueue (3);
```

Queue Implemented with a Linked List

```
dequeue ();
```

Finding a Path in a Maze (Non-recursive)

```
Place the starting square in a queue
while (the queue is not empty) {
  remove the front of the queue;
  for each neighbor of the removed square {
    if it has not been explored {
      if it is the destination {
        return true;
      }
      add it to the queue
    }
  }
}
```

Monday, April 22, 13
Queue Implemented with an Array

enqueue (1);
enqueue (2);
enqueue (3);

dqueue ();

dqueue is O(n)
Queue Implemented with an Array (Idea 2)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

front
enqueue (1);
enqueue (2);
enqueue (3);
enqueue (4);
enqueue (5);
enqueue (6);
enqueue (7);
enqueue (8);
dequeue ();

dequeue ();

Queue Implemented with an Array (Idea 2)

| 2 | 3 | 4 | 5 | 6 | 7 | 8 |

front
enqueue (1);
enqueue (2);
enqueue (3);
enqueue (4);
enqueue (5);
enqueue (6);
enqueue (7);
enqueue (8);
dequeue ();

dequeue ();
dequeue ();

Queue Implemented with an Array (Idea 2)

| 3 | 4 | 5 | 6 | 7 | 8 |

front
enqueue (1);
enqueue (2);
enqueue (3);
enqueue (4);
enqueue (5);
enqueue (6);
enqueue (7);
enqueue (8);
dequeue ();

Monday, April 22, 13
Queue Implemented with an Array (Idea 2)

- enqueue (1);
- enqueue (2);
- enqueue (3);
- enqueue (4);
- enqueue (5);
- enqueue (6);
- enqueue (7);
- enqueue (8);
- dequeue ();

Problem: We can only ever put n elements in the queue!

Queue Implemented with a CircularArray

- first
- last
- numItems = 3
- enqueue (1);
- enqueue (2);
- enqueue (3);
- dequeue ();

Queue Implemented with a CircularArray

- first
- last
- numItems = 2
- enqueue (1);
- enqueue (2);
- enqueue (3);
- dequeue ();
- enqueue (4);
Queue Implemented with a CircularArray

Enqueue 1;  enqueue 2;  enqueue 3;
numItems = 2
enqueue 4;  enqueue 5;  enqueue 6;  enqueue 7;  dequeue ();  dequeue ();  enqueue 8;

Enqueue 9;
numItems = 6
enqueue 1;  enqueue 5;  enqueue 6;  enqueue 7;  enqueue 8;  dequeue ();  dequeue ();  enqueue 8;
Queue Implemented with a CircularArray

<table>
<thead>
<tr>
<th>9</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>last</td>
<td>first</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

numItems = 7

enqueue (1); enqueue (5); enqueue (9);
enqueue (2); enqueue (6);
enqueue (3); enqueue (7);
dequeue (); dequeue ();
enqueue (4); enqueue (8);

Data Members for Circular Array

- int first; // index of front of queue
- int last;  // index of last element of queue
- int numItems; // how many in queue
- int capacity; // how many queue can hold
- T[] values;  // array for data

Queue in Collection Hierarchy

```
<<interface>>
 Iterable
   
<<interface>>
 Collection
   
<<interface>>
 Queue
   <<interface>>
 List
   <<abstract>>
 AbstractCollection
   
<<abstract>>
 AbstractList
   
LinkedList
```

Monday, April 22, 13