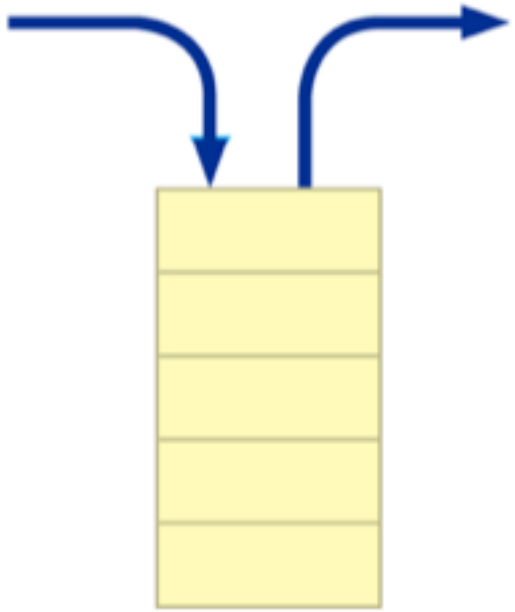
COSC 121: Computer Programming II

Dr. Bowen Hui

University of British Columbia Okanagan

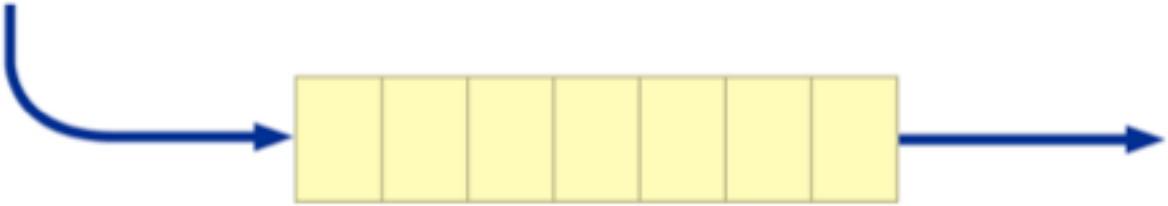
Review: Stacks

- Stack is a linear data structure
  - LIFO order
- Stack Operations:
  - push, pop, top, empty
- Last classes:
  - Implemented stack as array
  - Implemented stack as linked list



Review: Queues

- Queue is a linear data structure
  - FIFO order
- Queue Operations:
  - enqueue, dequeue, empty



Exercise: Implement Queue as an Array

- Problem specification:
  - An array of integers
  - Fixed input size
- Where to start in implementation?
  - Classes?
  - Methods?
  - What to keep track of?

Exercise: Implement Queue as an Array

- Classes?
  - Test class with main()
  - Queue class with its operations
- Methods?
  - enqueue(), dequeue()
  - is_empty() (may also want is_full())
  - toString()
- What to keep track of?
  - Whether queue is empty
  - Index to enqueue and dequeue

```
~/Documents/121/code$ cat QueueDemo.java
public class QueueDemo
{
  public static void main( String[] args )
  {
    Queue myqueue = new Queue( 5 );
    System.out.println( myqueue.is_full() );
    myqueue.enqueue( 4 );
    myqueue.enqueue( 2 );
    myqueue.enqueue( 9 );
    System.out.println( myqueue.toString() );
    System.out.println( myqueue.is_full() );
    myqueue.enqueue( 3 );
    myqueue.dequeue();
    myqueue.enqueue( 6 );
    System.out.println( myqueue.toString() );
    System.out.println( myqueue.is_full() );
    myqueue.enqueue( 1 );
    System.out.println( myqueue.is_full() );
    myqueue.enqueue( 0 );
    myqueue.enqueue( 0 );
    myqueue.enqueue( 0 );
    System.out.println( myqueue.is_full() );
    myqueue.dequeue();
    System.out.println( myqueue.is_full() );
    myqueue.dequeue();
    System.out.println( myqueue.toString() );
    myqueue.enqueue( 0 );
    System.out.println( myqueue.toString() );
  }
}
```

Output?

Output

```
~/Documents/121/code$ java QueueDemo
false
head pos = 0, tail pos = 3: 4 2 9 -1 -1
false
head pos = 1, tail pos = 0: -1 2 9 3 6
false
true
true
false
head pos = 3, tail pos = 1: 1 -1 -1 3 6
head pos = 3, tail pos = 2: 1 0 -1 3 6
~/Documents/121/code$
```

How to visualize stack operations with linked list?

Queue Class Skeleton

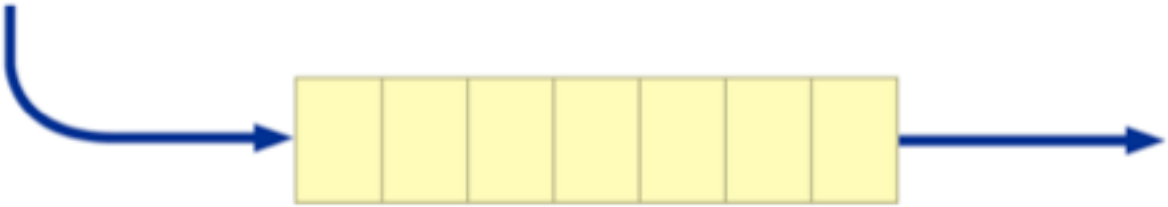
```
public class Queue
{
  // attributes

  // methods
  public Queue() { … }
  public boolean is_empty() { … }
  public void enqueue() { … }
  public int dequeue() { … }
  public String toString() { … }
}
```

Can have
is_full() as well

Exercise: Implement Queue as Linked List

- Problem specification:
  - A queue to store a set of integers
  - Size of list is variable
- Where to start in implementation?
  - Classes?
  - Methods?
  - What to keep track of?



Exercise: Implement Queue as Linked List

- Classes?
  - Test class with main()
  - Queue class with its operations
  - Node class to hold each element
- Methods?
  - enqueue(), dequeue()
  - is_empty() (may also want is_full())
  - toString()
- What to keep track of?
  - Whether queue is empty
  - Pointer to node “location” for enqueue and dequeue

```
public class QueueListDemo
{
  public static void main( String[] args )
  {
    QueueList myqueue = new QueueList();
    System.out.println( myqueue.toString() );
    myqueue.enqueue( 4 );
    myqueue.enqueue( 2 );
    myqueue.enqueue( 9 );
    myqueue.enqueue( 3 );
    System.out.println( myqueue.toString() );
    myqueue.dequeue();
    myqueue.dequeue();
    System.out.println( myqueue.toString() );
    myqueue.enqueue( 6 );
    System.out.println( myqueue.toString() );
    myqueue.enqueue( 1 );
    myqueue.enqueue( 0 );
    myqueue.enqueue( 0 );
    myqueue.enqueue( 0 );
    System.out.println( myqueue.toString() );
    myqueue.dequeue();
    myqueue.dequeue();
    System.out.println( myqueue.toString() );
    myqueue.enqueue( 0 );
    System.out.println( myqueue.toString() );
  }
}
```

Output?

Output

```
~/Documents/121/code$ java QueueListDemo

4   2   9   3
9   3
9   3   6
9   3   6   1   0   0   0
6   1   0   0   0
6   1   0   0   0   0
~/Documents/121/code$
```

How to visualize stack operations with linked list?

QueueNode Class Skeleton

```
public class QueueNode
{
  // attributes
  int item;
  QueueNode next;

  // methods
  public QueueNode( int val ) { … }
  public int getNext() { … }
  public void setNext( QueueNode n ) { … }
  public String toString() { … }
}
```

QueueList Class Skeleton

```
public class QueueList
{
   // attributes
   private QueueNode myqueue;
   private QueueNode tail;
   private final int UNDEF = -1;

   // methods
   public QueueList() { … }
   public boolean is_empty() { … }
   public void enqueue( int val ) { … }
   public int dequeue() { … }
   public String toString() { … }
}
```

Comparison of Implementations

- Stack
  - As an array – easier, but fixed size
  - As a list – extra work to fix pointers
- Queue
  - As an array – difficult to keep track of head and tail
  - As a list – easier to manage