Data Structures and Algorithms

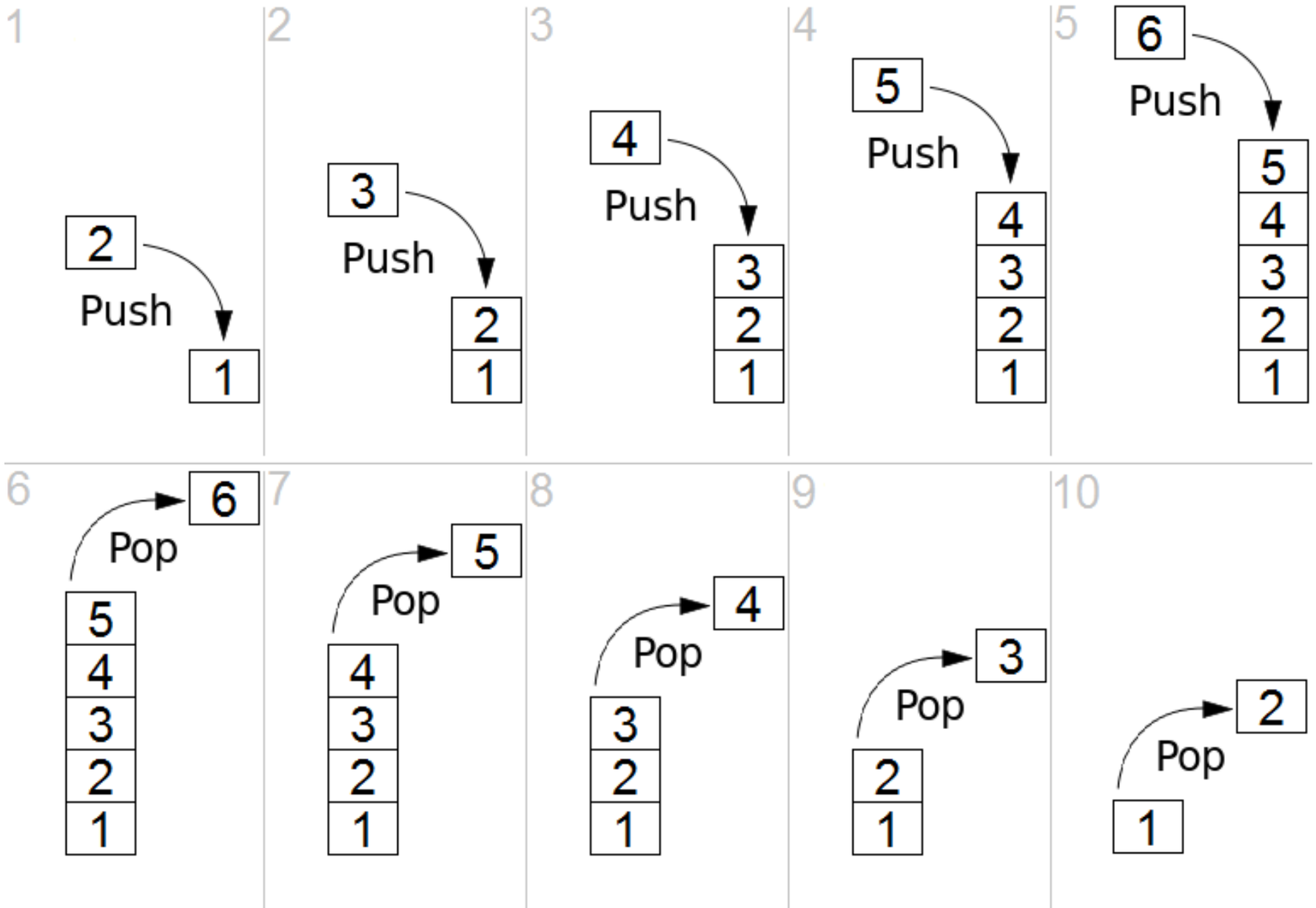
Dr Keith Maycock

Stack

Revision of last week

- Implementing a Stack
- Stack Pseudocode
- Interface

Stack


1
2
Push
1
2
3
Push
2
1
3
4
Push
3
2
1
4
5
Push
4
3
2
1
5
6
Push
5
4
3
2
1
6
6
Pop
5
4
3
2
1
7
5
Pop
4
3
2
1
8
4
Pop
3
2
1
9
3
Pop
2
1
10
2
Pop
1

Today

Topics Today

- Queues

Queue

Definition: An abstract data type (ADT) in which the entities in the collection are kept in order and the principal operations are the addition of entities to the rear terminal position, known as enqueue, and removal of entities from the front terminal position, known as dequeue.

This makes the queue a First-In-First-Out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed.

Often a peek operation is also supported, returning the value of the front element without dequeuing it.

Queue

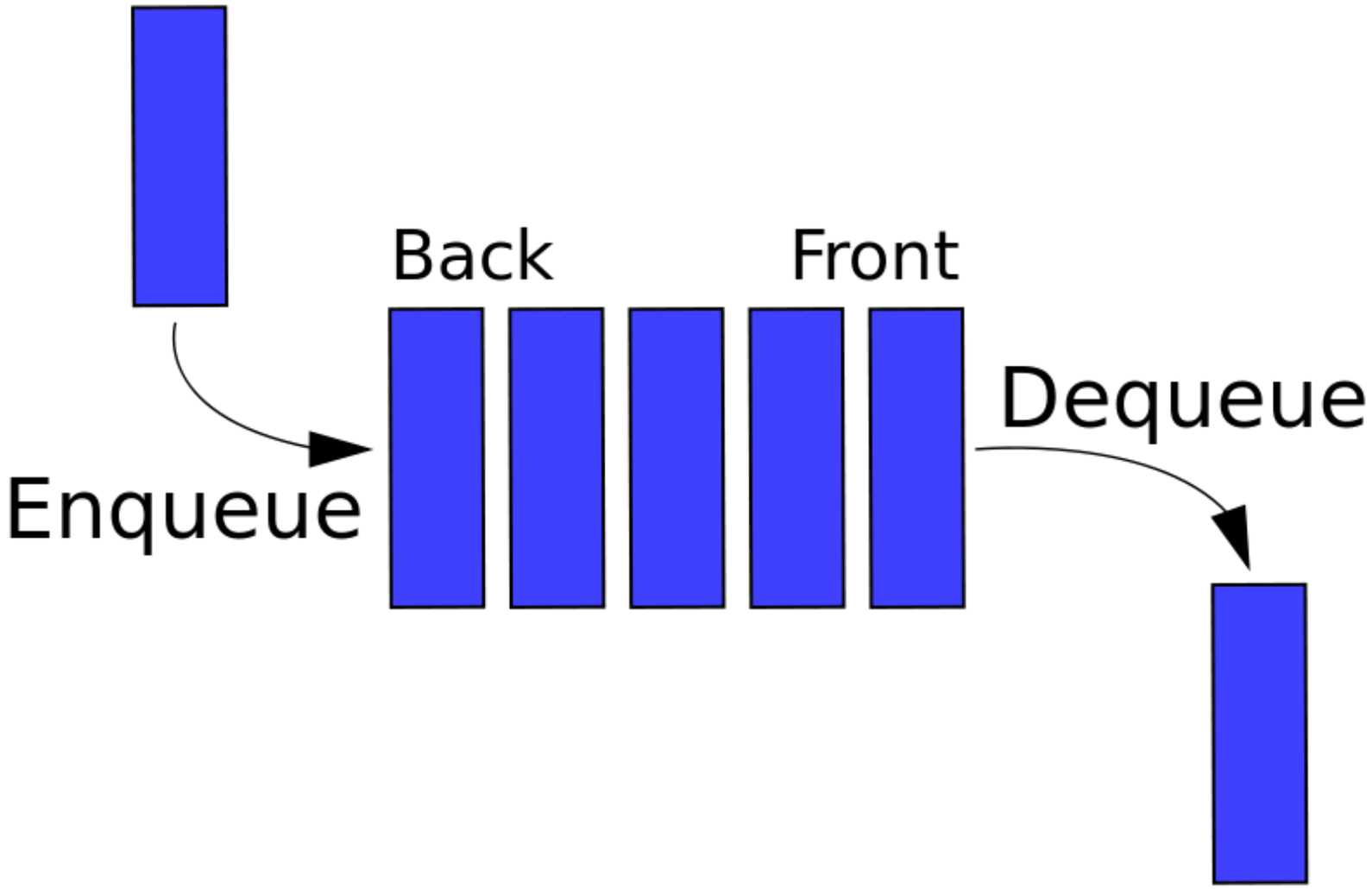


Figure: Queue ADT

Linked List Queue

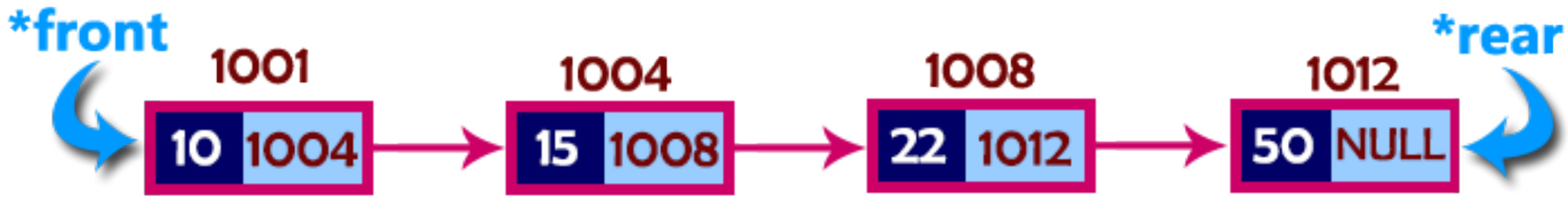


Figure: Linked List Queue

Queue Interface

```
public interface Queue<T>{
    public void enqueue(T elem);
    public T dequeue();
    public boolean isEmpty();
    public T peak();
    public int size();
}
```

Linked List Queue

Okay so we know the constructs…we need a Node class to use in our Linked List

```
public class Node<T>{
  T element;
  Node<T> next;

  public Node(T el, Node<T> n){
    element = el;
    next = n;
  }

  public Node(T el){
    element = el;
    next = null;
  }
```

ToDo: Complete the Node class and add a toString method that prints out the value of the element within the Node. Add a main method and create some Nodes.

Linked List Queue

Okay so we know the constructs…we need to create the Linked List variables and constructor

```
public class LinkedQueue<T>{
   private Node<T> first;
   private Node<T> last;

   public LinkedQueue(){
      first = null;
      last = null;
   }
```

Looks familiar? Can you explain whats happening here?

Linked List Queue

Lets look at the Interface again…

```
public interface Queue<T>{
  public void enqueue(T elem);
  public T dequeue();
  public boolean isEmpty();
  public T peak();
  public int size();
}
```

enqueue: adds an element at the end.

dequeue: removes an element from the queue

peak: method to return the first element at the top of the queue…without removing it.

Linked List Queue

Lets try and implement the Queue Interface without creating the methods and then compile…

```
Keiths-MacBook-Air:Queue kmaycock$ javac LinkedQueue.java
LinkedQueue.java:1: error: LinkedQueue is not abstract and does not override
abstract method size() in Queue
public class LinkedQueue<T> implements Queue<T>{
       ^
1 error
Keiths-MacBook-Air:Queue kmaycock$
```

ahhhhh: Java is not having fun here…what does this mean? Using google search this error message.

Linked List Queue

Lets create a custom error for when the queue is empty that prints out an error message for us…

```
public class EmptyQueueException extends RuntimeException{
  public EmptyQueueException(){
    super("There is nothing in the queue : )");
  }
}
```

EmptyQueueException: this custom error extends from RuntimeException and when called simply passes the string above, which is printed.You should save this in a file called "EmptyQueueException.java" in the same folder that you are creating your Linked List Queue

Linked List Queue

Create the method isEmpty()

- **You need to decide what test you are going to use to identify if the queue is empty?**
- **What are you going to return when you run your test?**
- **Wouldn't it be cool if you could just ask…is my queue empty and get a yes / no answer**

Linked List Queue

Implementing isEmpty()…

```
public boolean isEmpty(){
   return first == null;
}
```

test: checks if the first element is null

returns: a boolean variable gives use our yes or no answer.

Linked List Queue

Cool! We need to create the size method…

- **We need to return a variable that holds the size of the linked list.**
- **There are only two possible conditionals that we need to cater for. What are they?**
- **As we are working with a Linked List we need to be able to iterate through the list using a loop and incrementing our counter on each iteration**

Linked List Queue

Implementing size()…

```
public int size(){
  if (isEmpty()) {
    return 0;
  } else {
    int size = 0;
    Node<T> current = first;
    while (current != null) {
      size++;
      current = current.next;
    }
    return size;
  }
}
```

conditions: either the linked list is empty or its not.

size: we are using the variable size to hold the count of the number of elements.

while: this allows us to iterate through the linked list one element at a time.

Can you explain how the loop works?

Linked List Queue

Cool! We need to create the method Peek()…this lets us see the first element of the linked list…

- **There are only two possible conditionals that we need to cater for. What are they?**
- **If there is something in the Linked List we need to return a copy of the first element…the element that is at the top of the Queue**

Linked List Queue

Implementing peek()…

```
public T peak(){
  if(isEmpty()){
    throw new EmptyQueueException();
  }
  else{
    return first.element;
  }
}
```

conditions: either the linked list is empty or its not.

return: as we are returning the data from the first element, this is accessed through first.element

Can you explain how the loop works?

Oops, can you see an error in the method signature?

Linked List Queue

Okay so we are getting close…we need to be able to add an element to the queue…called enqueue

- **There are only two possible conditionals that we need to cater for. What are they?**
- **We need to consider what we are adding and then add this to the end of the queue…what is our reference Node for the end?**

Linked List Queue

Creating enqueue…

```
public void enqueue(T elem){
  Node<T> oldlast = last;
  last = new Node<T>(elem);
  if(isEmpty()){
    first = last;
  }else{
    oldlast.next = last;
  }
}
```

conditions: either the linked list is empty or its not.

Node: we are simply trying to add a new Node to our linked list queue. As its a queue the node goes at the end of the queue.

Can you explain how the loop works?

Linked List Queue

Okay so we are getting close…we need to be able to remove an element to the queue…called dequeue

- **There are only two possible conditionals that we need to cater for. What are they?**

- **We need to consider what we want to return from the queue.**

- **We should consider how a queue works and what element should be returned when removing an element.**

Linked List Queue

Creating dequeue…

```
public T dequeue(){
  if(isEmpty()){
    throw new EmptyQueueException();
  }
  T elem = first.element;
  first = first.next;
  if(isEmpty()){
    last = null;
  }
  return elem;
}
```

conditions: either the linked list is empty or its not. If it is we throw our new exception.

Node: we are simply trying to remove the first Node. When we do we need to set the last Node if we have removed the only Node in the queue

Can you explain how the loop works?

Linked List Queue

Like the other data structures we should be able to print out all the elements…

- **There are only two possible conditionals that we need to cater for. What are they?**
- **We need to consider what we want to return from the queue.**
- **We need all the elements…**

Linked List Queue

Creating toString…

```
public String toString(){
  String output = "";
  if(isEmpty()){
      output = output + "There are no elements in the LinkedQueue";
  }
  else{
    Node<T> temp = first;
    while(temp != null){
      output = output + temp.element + " : ";
      temp = temp.next;
    }
  }
  return output;
}
```

Can you explain how this code works?

Linked List Queue

Great. You have have the Queue implemented. Complete the following

- **Create an instance of a new Linked List Queue of type Integer**
- **Add three ints into your queue**
- **Print out the size of your queue**
- **Remove two elements from your queue**
- **Re-implement a queue using an array instead of a linked list**

The End