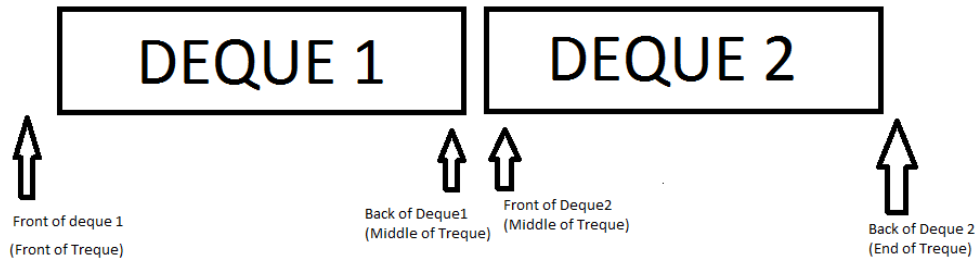
I have decided to make a new Deque class which had the same methods as the ArrayDeque. The Treque class will implement the Deque class by initializing two Deques (the leftDeque and the rightDeque). These four ends will act as the 3 access points of the Treque. However, the Treque class will override the get, set, add, remove and implement its own balance function. Adding and removing from the middle of the list will be constant because you are really adding/removing from both ends of the two deques that make up the Treque (which is always constant time).



TREQUE COMPOSITION

The **get** method will cycle through the list and return the element:

```
public T get(int i)
  {
    if( i < 0 || i > size() ) throw new IndexOutOfBoundsException();


    if(i < leftDeque.size())
       return leftDeque.get(i);
    else
       return rightDeque.get(i - leftDeque.size());
  }
```

The **set** method will set the value at an index to the one in the parameter:

```
public T set(int i, T x)
        {
                T y;
                if( i < 0 || i > size() ) throw new IndexOutOfBoundsException();

                if(i < leftDeque.size())
      y = leftDeque.set(i,x);
    else
      y = rightDeque.set(i - leftDeque.size(), x);

        return y;
        }
```

The **add** function will add an element based on the index

```
public void add(int i, T x)
  {
    if ( i < 0 || i > size() )
      throw new IndexOutOfBoundsException();

    if (i < leftDeque.size())
      leftDeque.add(i, x);
    else
      rightDeque.add(i - leftDeque.size(), x);

    balance();   //call balance everytime an element is added
  }
```

The **remove** function will remove an element based on the index

```
public T remove(int i)
  {
     T x;
     if( i < 0 || i > size() )
        throw new IndexOutOfBoundsException();

     if(i < leftDeque.size())
        x = leftDeque.remove(i);
     else
        x = rightDeque.remove(i - leftDeque.size());

     balance();   //call balance every time an element is remove
     return x;
  }
```