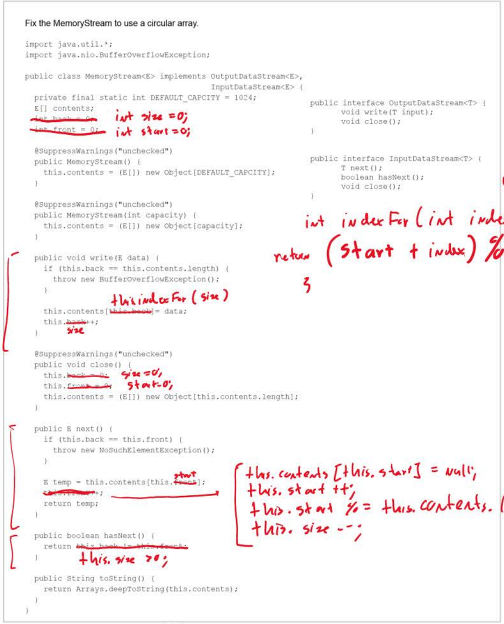
PA7 due Tuesday
PA8 due week 10 (Tuesday)



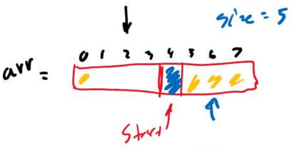
Fix the MemoryStream to use a circular array.

```
import java.util.*;
import java.nio.BufferOverflowException;

public class MemoryStream<E> implements OutputDataStream<E>,
                                        InputDataStream<E> {
  private final static int DEFAULT_CAPCITY = 1024;
  E[] contents;
  ~~int back = 0;~~   int size =0;
  ~~int front = 0;~~   int start =0;

  @SuppressWarnings("unchecked")
  public MemoryStream() {
    this.contents = (E[]) new Object[DEFAULT_CAPCITY];
  }

  @SuppressWarnings("unchecked")
  public MemoryStream(int capacity) {
    this.contents = (E[]) new Object[capacity];
  }

  public void write(E data) {
    if (this.back == this.contents.length) {
      throw new BufferOverflowException();
    }
                      this.indexFor ( size )
    this.contents[~~this.back~~]= data;
    this.~~back~~++;
         size
  }

  @SuppressWarnings("unchecked")
  public void close() {
    this.~~back = 0;~~   size =0;
    this.~~front = 0;~~   start=0;
    this.contents = (E[]) new Object[this.contents.length];
  }

  public E next() {
    if (this.back == this.front) {
      throw new NoSuchElementException();
    }
                                       start
    E temp = this.contents[this.~~front~~];
    ~~this.front++;~~
    return temp;
  }

  public boolean hasNext() {
    return ~~this.back != this.front;~~
           this.size >0;
  }

  public String toString() {
    return Arrays.deepToString(this.contents);
  }
}
```

```
public interface OutputDataStream<T> {
      void write(T input);
      void close();
}

public interface InputDataStream<T> {
      T next();
      boolean hasNext();
      void close();
}
```

```
int indexFor (int index) {
  return (start + index) % this.contents.length;
}
```

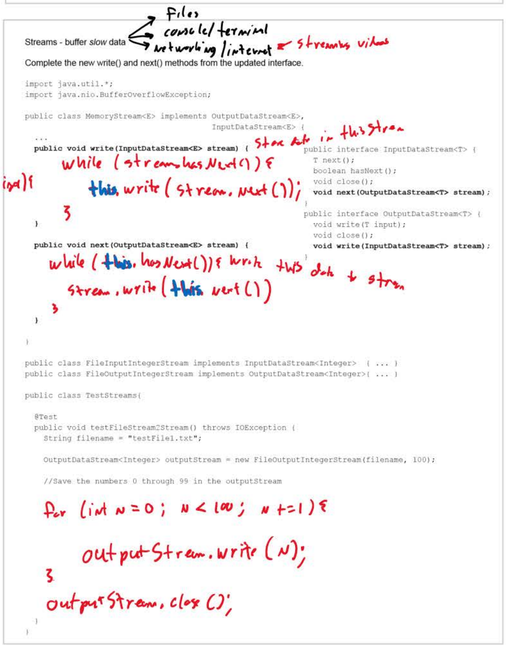
```
this.contents [this.start] = null;
this.start ++;
this.start %= this.contents.length; → this.start = this.start %
                                                  this.contents.length
this.size --;
```



~~arr[2]~~

arr[indexFor (2)]

start +2
4 +2 =6
[6]



Streams - buffer *slow* data → Files / console/terminal / networking/internet ← streaming videos

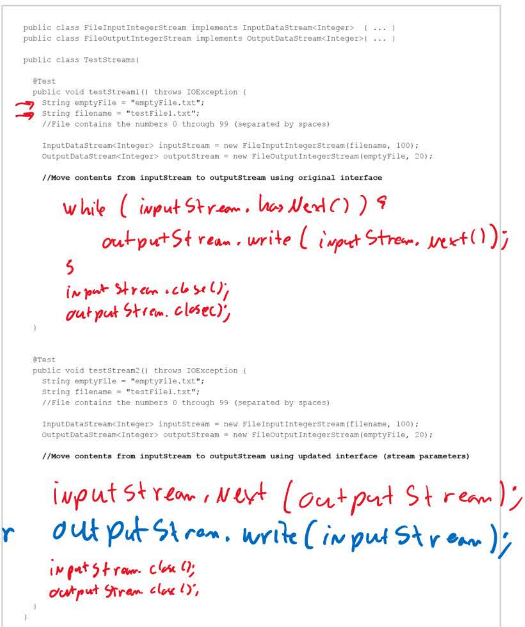
Complete the new write() and next() methods from the updated interface.

```
import java.util.*;
import java.nio.BufferOverflowException;

public class MemoryStream<E> implements OutputDataStream<E>,
                                        InputDataStream<E> {
  ...
  public void write(InputDataStream<E> stream) {   store data in this stream
    while (stream.hasNext()) {
      this.write(stream.next());
    }
  }

  public void next(OutputDataStream<E> stream) {
    while (this.hasNext()) {   write this data to stream
      stream.write(this.next())
    }
  }

}
```

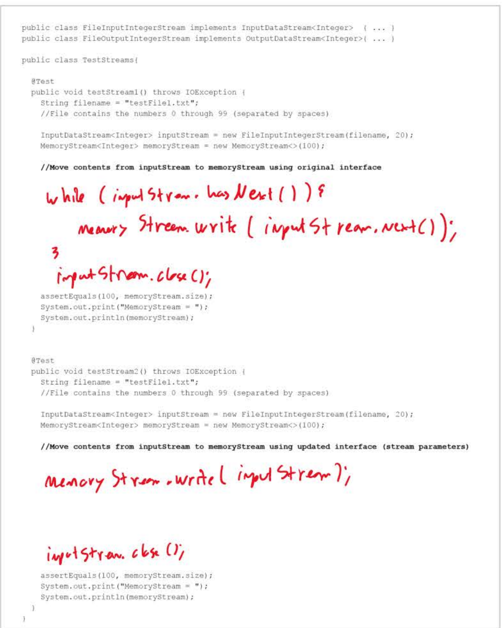
MemoryStream (Input Data Stream input) {
}

```
public interface InputDataStream<T> {
  T next();
  boolean hasNext();
  void close();
  void next(OutputDataStream<T> stream);
}
public interface OutputDataStream<T> {
  void write(T input);
  void close();
  void write(InputDataStream<T> stream);
}
```

```
public class FileInputIntegerStream implements InputDataStream<Integer>  { ... }
public class FileOutputIntegerStream implements OutputDataStream<Integer>{ ... }

public class TestStreams{

  @Test
  public void testFileStream2Stream() throws IOException {
    String filename = "testFile1.txt";

    OutputDataStream<Integer> outputStream = new FileOutputIntegerStream(filename, 100);

    //Save the numbers 0 through 99 in the outputStream

    for (int N = 0; N < 100; N +=1) {

        outputStream.write(N);
    }
    outputStream.close();
  }
}
```



```
public class FileInputIntegerStream implements InputDataStream<Integer>  { ... }
public class FileOutputIntegerStream implements OutputDataStream<Integer>{ ... }

public class TestStreams{

  @Test
  public void testStream1() throws IOException {
→   String emptyFile = "emptyFile.txt";
→   String filename = "testFile1.txt";
    //File contains the numbers 0 through 99 (separated by spaces)

    InputDataStream<Integer> inputStream = new FileInputIntegerStream(filename, 100);
    OutputDataStream<Integer> outputStream = new FileOutputIntegerStream(emptyFile, 20);

    //Move contents from inputStream to outputStream using original interface

      while ( inputStream.hasNext() ) {
          outputStream.write( inputStream.next());
      }
      inputStream.close();
      outputStream.close();
  }

  @Test
  public void testStream2() throws IOException {
    String emptyFile = "emptyFile.txt";
    String filename = "testFile1.txt";
    //File contains the numbers 0 through 99 (separated by spaces)

    InputDataStream<Integer> inputStream = new FileInputIntegerStream(filename, 100);
    OutputDataStream<Integer> outputStream = new FileOutputIntegerStream(emptyFile, 20);

    //Move contents from inputStream to outputStream using updated interface (stream parameters)

     inputStream.Next (outputStream);
or   outputStream.write(inputStream);
     inputStream.close();
     outputStream.close();
  }
}
```



```
public class FileInputIntegerStream implements InputDataStream<Integer>  { ... }
public class FileOutputIntegerStream implements OutputDataStream<Integer>{ ... }

public class TestStreams{

  @Test
  public void testStream1() throws IOException {
    String filename = "testFile1.txt";
    //File contains the numbers 0 through 99 (separated by spaces)

    InputDataStream<Integer> inputStream = new FileInputIntegerStream(filename, 20);
    MemoryStream<Integer> memoryStream = new MemoryStream<>(100);

    //Move contents from inputStream to memoryStream using original interface

    while (inputStream.hasNext()) {
        memoryStream.write ( inputStream.next());
    }
     inputStream.close();
    assertEquals(100, memoryStream.size);
    System.out.print("MemoryStream = ");
    System.out.println(memoryStream);
  }

  @Test
  public void testStream2() throws IOException {
    String filename = "testFile1.txt";
    //File contains the numbers 0 through 99 (separated by spaces)

    InputDataStream<Integer> inputStream = new FileInputIntegerStream(filename, 20);
    MemoryStream<Integer> memoryStream = new MemoryStream<>(100);

    //Move contents from inputStream to memoryStream using updated interface (stream parameters)

    memoryStream.write( inputStream);


    inputStream.close();
    assertEquals(100, memoryStream.size);
    System.out.print("MemoryStream = ");
    System.out.println(memoryStream);
  }
}
```