

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

```
public class MemoryStream<E> implements OutputStream<E>,
        InputStream<E> {
    private final static int DEFAULT_CAPACITY = 1024;
    E[] contents;
    int size = 0;
    int start = 0;

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

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

    public int capacity() {
        return this.contents.length;
    }

    public int size() {
        return this.size;
    }

    private int indexFor(int index) {
        return (start + index) %this.capacity();
    }

    public void write(E data) {
        if (this.size() >= this.capacity()) {
            //ArrayList? expandCapacity();
            throw new BufferOverflowException();
        }

        this.contents[this.indexFor(this.size)] = data;
        this.size++;
    }
}
```

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

public interface OutputStream<T> {
    void write(T input);
    void close();
    void write(InputStream<T> stream);
}
```

```
public void close() {
    this.start = 0;
    this.size = 0;
    Arrays.fill(this.contents, null);
}

public E next() {
    if (this.size == 0) {
        throw new NoSuchElementException();
    }

    E temp = this.contents[this.start];
    this.contents[this.start] = null;
    this.start++;
    this.start %= this.capacity();
    this.size--;

    return temp;
}

public boolean hasNext() {
    return this.size > 0;
}

public void write(        InputStream    <E> stream) {
}

public void next(        OutputStream    <E> stream) {
}

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