```java
import java.util.concurrent.Semaphore;

/**
 * Buffer circular com uso de Semáforos para região crítica e
sincronização entre produtores e consumidores.
 */
public class BufferComMutexSincronizacaoComSemaforos implements
Buffer {
   private String[] itens;
   private int quantidade = 0;
   private int inicio = 0;
   private int fim = 0;

   private Semaphore mutex;
   private Semaphore cheio;
   private Semaphore livre;

   public BufferComMutexSincronizacaoComSemaforos(int capacidade) {
      this.itens = new String[capacidade];
      cheio = new Semaphore(0);           // nenhuma posição cheia
      livre = new Semaphore(capacidade); // todas as posições livres
      mutex = new Semaphore(1);          // mutex binário
   }

   public void addItem(String item) throws Exception {
      try {
         // só prossegue se há posição livre (livre.s > 0)
         livre.acquire();
         mutex.acquire();

         // se buffer cheio, então retorna exceção
         if (quantidade == itens.length) {
            throw new Exception("Buffer Cheio");
         }
         // senão, adiciona no final e
         // atualiza indice e quantidade
         itens[fim] = item;
         fim = (fim + 1) % itens.length;
         quantidade++;
         print();

      } catch (InterruptedException e) {
         System.out.println("semaforo mutex ou livre interrompidos");
         e.printStackTrace();
      } finally {
         mutex.release(); // ao final, liberar mutex e
         cheio.release(); // liberar consumidor que estava aguardando
      }
   }
```

```java
public interface Buffer {
    public void addItem(String item) throws Exception;

    public String removeItem() throws Exception;
}
```

```java
   public String removeItem() throws Exception {
      String item = null;
      try {
         // só prossegue se há posição ocupada (cheio.s > 0)
         cheio.acquire();
         mutex.acquire();

         // se buffer vazio, então retorna exceção
         if (quantidade == 0) {
            throw new Exception("Buffer Vazio");
         }
         // senão, remove do início e
         // atualiza indice e quantidade
         item = itens[inicio];
         itens[inicio] = null;
         inicio = (inicio + 1) % itens.length;
         quantidade--;
         print();
      } catch (InterruptedException e) {
         System.out.println("semaforo mutex ou livre interrompidos");
         e.printStackTrace();
         return null;
      } finally {
         mutex.release(); // ao final, liberar o mutex e
         livre.release(); // liberar produtor que estava aguardando
      }
      return item;
   }
}
```