Exercícios: Filas

1. Considere o trecho de código abaixo, onde `f` é uma fila com capacidade para 5 elementos e que está inicialmente vazia:

```
public void foobar(Queue<Integer> f) {
    try {
         f.enqueue(1);
         f.enqueue(3);
         f.enqueue(5);
         f.enqueue(7);
         f.enqueue(9);
    } catch (OverflowException e) {
         System.out.println(e);
    }
    try {
         f.dequeue();
         f.dequeue();
    } catch (UnderflowException e) {
         System.out.println(e);
    }
    try {
         f.enqueue(2);
         f.enqueue(4);
    } catch (OverflowException e) {
         System.out.println(e);
    }
    try {
         f.dequeue();
    } catch (UnderflowException e) {
         System.out.println(e);
    }
    try {
         f.enqueue(3);
    } catch (OverflowException e) {
         System.out.println(e);
    }
      System.out.println(f);
}
```

Assinale V ou F para os itens a seguir:

( ) A ordem final dos elementos na fila (último println executado) é {2, 4, 3, 7, 9};

( ) Não acontece exceção alguma no código acima.

2. Implemente, em uma classe qualquer, um método que recebe duas filas `q1` e `q2` de inteiros e transfere os elementos da segunda para a primeira de modo que eles fiquem na frente dos originais. Exemplo: supondo `q1` = {1, 2, 3, 4} e `q2` = {5, 6, 7, 8}, após

a chamada do método devemos ter **`q1`** = {5, 6, 7, 8, 1, 2, 3, 4} e **`q2`** = {}. Dica: use uma fila auxiliar.

```
public void prependQueue(Queue<Integer> q1, Queue<Integer> q2)
```

3. Implemente, em uma classe qualquer, um método que remove todas as ocorrências de um determinado elemento em uma fila de caracteres. Considere a assinatura a seguir:

```
public void exterminateFromQueue(Queue<Character> q, char element)
```

Nos exercícios a seguir, os métodos solicitados devem ser implementados dentro das classes que implementam filas. Salvo indicação em contrário, as estruturas passadas como parâmetro (arrays, filas, etc.) devem ser preservadas, ou seja, seus elementos não devem ser removidos ou trocados de ordem.

4. Implemente o método **`contains`**, definido abaixo, que informa se a fila contém determinado elemento.

```
public boolean contains(E element)
```

5. Implemente um método que inverte a ordem dos elementos da fila.

```
public void flip()
```

6. Implemente uma sobrecarga do método **`enqueue`** que recebe como parâmetro uma fila, em vez de um elemento. Esse método deve adicionar ao final da fila corrente os elementos da fila passada como parâmetro, mantendo a ordem original.

7. Implemente o método abaixo, que insere um elemento na primeira posição da fila.

```
public void enqueueWithPriority(E element)
```

8. Implemente um método **`equals`** para a fila. Uma fila será igual a outra se contiver os mesmos elementos, dispostos na mesma ordem. Para comparar os elementos, use também o método **`equals`**.

9. Implemente um método **`clone`** para a fila. Esse método deve retornar uma nova fila contendo os mesmos elementos da atual. Os elementos em si não devem ser duplicados.

10. Considerando a classe **`StaticQueue`**, mostre como ficaria o array interno (**`elements`**) do objeto **`q`** ao fim da execução do código a seguir:

```
public static void main(String[] args) {
      StaticQueue<Integer> q = new StaticQueue<>(5);
      try {
            q.enqueue(9);
            q.enqueue(8);
            q.enqueue(5);
            q.enqueue(2);
            q.dequeue();
            q.dequeue();
            q.enqueue(1);
            q.enqueue(2);
            q.enqueue(3);
            q.enqueue(7);
      }catch (Exception e) {
            System.err.println(e.toString());
      }
}
```

11. Implemente o método **`split`**, de acordo com a assinatura definida a seguir, que divide a fila em duas partes. A fila corrente deve permanecer somente com os elementos do início até a primeira ocorrência de **`element`**, inclusive. O restante dos elementos deve ser retornado em uma nova fila. Para comparar os elementos, utilize o método **`equals`**.

```
public Queue<E> split(E element)
```

12. Implemente um método que percorra a fila e mova para o final todas as ocorrências de um determinado elemento.

```
public void moveToBackAllOccurrencesOf(E element)
```

13. Implemente na classe **`StaticQueue`** um método que aumente a capacidade de armazenamento da fila, se necessário, para o valor passado como parâmetro. Os elementos atuais devem ser preservados.

```
public void ensureCapacity(int capacity)
```