Aufgabe 6.1: Arrays

6.1.1 `int`-Array als Stack

Die Klasse `Stack` soll ein `int`-Array kapseln, welches als Stack dienen soll. Die Funktionsweise eines Stacks wird in Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.** erklärt. Für den Zugriff auf den Stack sollen die Methoden

```
public void push (int u)
public int pop()
```

bereitgestellt werden.

Die Methode

```
public boolean isEmpty()
```

überprüft, ob der Stack leer ist, und liefert in diesem Fall `true` zurück, ansonsten wird `false` zurückgeliefert. Die Methode

```
public void stackPrint()
```

soll zu Testzwecken dienen und den Inhalt des gesamten Stacks ausgeben. Die Größe des Stacks soll dem Konstruktor übergeben werden können. Testen Sie die Klasse `Stack` mit Hilfe der folgenden Wrapper-Klasse:

```
// Datei: TestStack.java

public class TestStack
{
   public static void main (String[] args)
   {
      Stack stackRef = new Stack (5);
      stackRef.push (7);
      stackRef.push (3);
      stackRef.push (4);
      stackRef.push (9);
      stackRef.push (1);

      stackRef.stackPrint();

      System.out.println ("\nAusgabe der Zahlen: ");
      while (stackRef.isEmpty() == false)
      {
         int rückgabe;

         // oberste Zahl des Stacks wird
         // mit pop() vom Stack geholt
         rückgabe = stackRef.pop();
         System.out.println ("Die Zahl war " + rückgabe);
      }
   }
}
```

6.1.2 Array mit einfachen Datentypen – die Klasse `FloatQueue`

Die selbst geschriebene Klasse `FloatQueue` ist eine Warteschlange für `float`-Werte. In dieser Warteschlange können sich mehrere `float`-Werte befinden. Es kann jeweils nur ein Element gleichzeitig in die Warteschlange (hinten) eingereiht werden (`enqueue()`-Methode) oder aus der Warteschlange (vorne) entnommen werden (`dequeue()`-Methode). Im Gegensatz zu einem Stapelspeicher (Stack) handelt es sich bei einer Warteschlange um einen **FIFO**-Speicher ("**First In – First Out**").

Die Klasse `FloatQueue` soll folgende Methoden beinhalten:

- **Konstruktor: `public FloatQueue (int laenge)`**

  Der Übergabeparameter `int laenge` gibt die Anzahl der maximalen Speicherstellen der Warteschlange an.

- **In Warteschlange einfügen: `public void enqueue (float wert)`**

  Diese Methode fügt den Wert am Ende der Warteschlange ein.

- **Aus Warteschlange entnehmen: `public float dequeue()`**

  Diese Methode entfernt den ersten Wert aus der Warteschlange und gibt diesen an den Aufrufer zurück. Ist die Warteschlange leer, so wird der Wert `-1` zurückgegeben.

- **Ausgabe des Inhalts der Warteschlange: `public void queuePrint()`**

  Diese Methode gibt alle in der Warteschlange enthaltenen Werte aus.

- **Überprüfen, ob die Warteschlange leer ist: `public boolean isEmpty()`**

  Diese Methode liefert `true` zurück, falls die Warteschlange leer ist. Andernfalls gibt die Methode `false` zurück.

- **Leeren der Warteschlange: `public void clear()`**

  Diese Methode löscht alle in der Warteschlange enthaltenen Werte.

Testen Sie die Klasse `FloatQueue` mit Hilfe folgender Testklasse:

```
// Datei: TestFloatQueue.java

public class TestFloatQueue
{
   public static void main (String[] args)
   {
      FloatQueue queue = new FloatQueue(5);
      queue.enqueue (2.45f);
      queue.enqueue (1.29f);
      queue.enqueue (4.31f);
      queue.enqueue (7.85f);

      queue.queuePrint();

      System.out.println ("\nAusgabe der Zahlen: ");

      while (queue.isEmpty() == false )
```

```
        {
            float rueckgabe;
            rueckgabe = queue.dequeue();
            System.out.println ("Die Zahl war " + rueckgabe);
        }

        queue.enqueue (1.11f);
        queue.queuePrint();
        queue.clear();
        queue.queuePrint();
    }
}
```