Übungsblatt 10 (25 Punkte)

10.1 Klasse Liste (15 Punkte)

Gegeben sei die Klasse Node und eine Schnittstelle `AbstrakteListe`.

Schreiben Sie eine Java-Klasse `Liste`, die diese Schnittstelle implementiert und unter Verwendung der Klasse Node eine *einfach verkettete* Liste modelliert. Kommentieren Sie zusätzlich einzelne (nichttriviale Schritte der) Methoden der Klasse `Liste`. Bei Bedarf können Sie weitere Methoden in dieser Klasse implementieren.

Implementieren Sie in der Klasse `Liste` die `toString()`-Methode, so dass eine Liste 1, 2, 3, 4 wie folgt auf dem Bildschirm ausgegeben wird:

```
[ 1 2 3 4 ]
```

10.2 Klasse Stapel (5 Punkte)

Die Klasse `Stapel` soll über folgende Methoden verfügen:

- `push()`: legt ein Element (mit einem gegebenen Wert) oben auf den Stack
- `pop()`: nimmt das oberste Element vom Stack herunter und gibt seinen Wert zurück
- `top()`: liefert den Wert des obersten Elements des Stacks zurück
- `isEmpty()`: liefert true zurück, wenn der Stack leer ist

Implementieren Sie eine entsprechende Schnittstelle `StapelSchnittstelle`, welche die o.g. Methoden beinhaltet, und kommentieren Sie diese Schnittstellenmethode im Javadoc-Format. Implementieren Sie eine Klasse `Stapel`, die von der Klasse `Liste` abgeleitet ist und die Schnittstelle `StapelSchnittstelle` implementiert.

Implementieren Sie in der Klasse `Stapel` die `toString()`-Methode, so dass ein Stapel 1, 2, 3, 4 wie folgt auf dem Bildschirm ausgegeben wird (die unterste *-Reihe markiert den Boden eines Stapels und muss nicht unbedingt an die Breite der vorangehenden Zeile angepasst werden):

```
** 4 **
** 3 **
** 2 **
** 1 **
*******
```

10.3 Klasse Warteschlange (5 Punkte)

Die Klasse `Warteschlange` soll über folgende Methoden verfügen:

- `enqueue()`: fügt ein Element (mit einem gegebenen Wert) ans Ende der Warteschlange hinzu
- `dequeue()`: entnimmt das erste Element aus der Warteschlange und liefert seinen Wert zurück
- `first()`: liefert den Wert des ersten Elements der Warteschlange zurück
- `isEmpty()`: liefert true zurück, wenn die Warteschlange leer ist

Implementieren Sie eine entsprechende Schnittstelle `WarteschlangeSchnittstelle`, welche die o.g. Methoden beinhaltet, und kommentieren Sie diese Schnittstellenmethode im Javadoc-Format.

Implementieren Sie eine Klasse `Warteschlange`, die von der Klasse `Liste` abgeleitet ist und die Schnittstelle `WarteschlangeSchnittstelle` implementiert.

Implementieren Sie in der Klasse Warteschlange die `toString()`-Methode, so dass eine Warteschlange 1, 2, 3, 4 wie folgt auf dem Bildschirm ausgegeben wird:

```
<--- 1 2 3 4 <---
```

Allgemeine Hinweise:

Achten Sie bei den Klassen `Stapel` und `Warteschlange` auf eine effiziente Implementierung, die die Vererbungsbeziehung zu der Klasse `Liste` ausnutzt und die dort bereitgestellten Methoden verwendet.

Verwenden Sie keine vordefinierten Java-Methoden, bis auf Ausgabe-Methoden, wie `System.out.print()` und – falls bei der Implementierung der `toString()`-Methoden nötig – die Methoden der Java-Klasse `String`.

Bennen Sie alle Methoden in den Klassen so, wie bei den Aufgaben vorgegeben.