Name: _____ Mat.Nr.: _____

1 Aufgabe

Implementieren Sie die Methode **void** sucheKuerzestenWeg(**int** start, **int** ziel) in der Klasse Orts-Liste, sodass diese folgende Aufgabe erfüllt:

1. Die Methode berechnet den kürzesten Weg zwischen dem `Ort start` und dem `Ort ziel`.
2. Dabei darf maximal eine Zwischenstation passiert werden.
3. Die Methode soll die Endergebnisse (Gesamtlänge, Zwischenstation) mit System.out.println (...) auf das Ausgabefenster schreiben.

2 Klassendefinitionen

2.1 Klasse OrtsListe

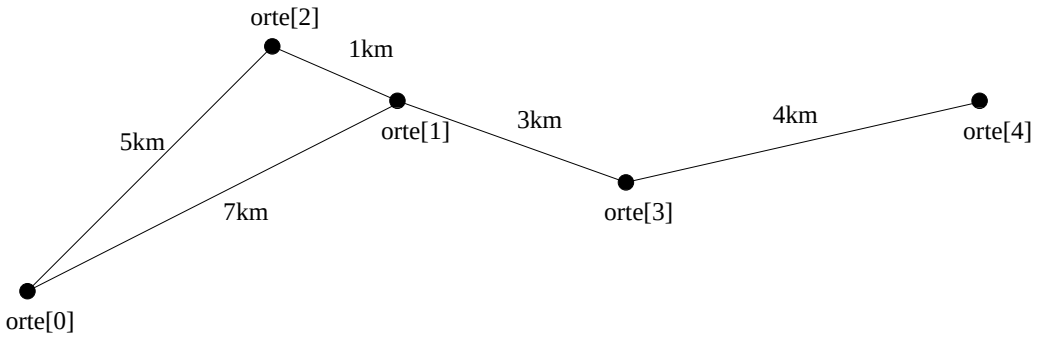
```
/**
 * Verwaltet eine Liste von Orten.
 */
public class OrtsListe {

    /**
     * Liste der Orte
     */
    Ort[] orte;

    /**
     * Constructor: Speichert eine Liste von Orten.
     */
    public OrtsListe(Ort[] liste) {
        orte = liste;
    }

    void sucheKuerzestenWeg(int start, int ziel) {
        ...
    }
}
```

2.2 Klasse Ort

```
/**
 * Repraesentiert einen Ort mit seinen Strassen zu Nachbarorten.
 */
public class Ort {

    /**
     * istVerbunden[i] gibt an, ob eine Verbindung zum Ort i besteht.
     */
    boolean[] istVerbunden;

    /**
     * Wenn istVerbunden[i] == true, dann gibt strassenLaenge[i] die
     * Laenge dieser Verbindung an.
     */
    int[] strassenLaenge;

    /**
     * Constructor: Legt einen neuen Ort mit seinen Verbindungen an.
     */
    public Ort(boolean[] verbunden, int[] strassenkm) {
        istVerbunden   = verbunden;
        strassenLaenge = strassenkm;
    }

    /**
     * Die Methode prueft, ob von diesem Ort eine direkte Verbindung
     * mit dem Zielort "ortnummer" besteht. Sie liefert "true"
     * zurueck falls die Verbindung existiert.
     */
    boolean istVerbundenMit(int ortnummer) {
        return(istVerbunden[ortnummer]);
    }

    /**
     * Liefert die Laenge der Verbindung zum Zielort "ortnummer"
     * zurueck.
     */
    int getKmZuOrt(int ortnummer) {
        return(strassenLaenge[ortnummer]);
    }

}
```

3 Beispiel

Das Beispiel zeigt fünf Orte die mit Straßen verbunden sind.



Man beachte die direkte Bergstraße von Ort `orte[0]` zu Ort `orte[1]`. Diese Route ist länger als die Route über den Ort `orte[2]`. Der Algorithmus soll daher den Weg über den Ort `orte[2]` vorschlagen und diese Weglänge berechnen.
Testaufrufe:

```
ortliste.sucheKuerzestenWeg(0, 2);
ortliste.sucheKuerzestenWeg(0, 1);
ortliste.sucheKuerzestenWeg(0, 3);
ortliste.sucheKuerzestenWeg(0, 4);
```

Für die obigen Aufrufe **könnte** folgende Ausgabe am Bildschirm erscheinen:

```
Ausgangspunkt der Reise: Ort 0
Zielort der Reise: Ort 2
Der kürzeste Weg ist eine Direktverbindung und beträgt 5km.
------------------------------------------------------------------------
Ausgangspunkt der Reise: Ort 0
Zielort der Reise: Ort 1
Die kürzeste Weg erfolgt über den Ort 2 und beträgt 6km.
------------------------------------------------------------------------
Ausgangspunkt der Reise: Ort 0
Zielort der Reise: Ort 3
Die kürzeste Weg erfolgt über den Ort 1 und beträgt 10km.
------------------------------------------------------------------------
Ausgangspunkt der Reise: Ort 0
Zielort der Reise: Ort 4
Es existiert keine Route mit maximal einer Zwischenstation!
------------------------------------------------------------------------
```

```
/**
 * Sucht den kuerzesten Weg fuer eine Reise zwischen zwei Orten,
 * wobei hoechstens ein Ort als Zwischenstation besucht wird.
 * Falls es keine solche Route gibt, soll dieser Umstand ausgegeben
 * werden. Andernfalls soll die Gesamtlaenge der Route und die
 * moeglicherweise enthaltene Zwischenstation ausgegeben werden.
 */
void sucheKuerzestenWeg(int start, int ziel) {
```