In den Zeilen 5 bis 14 werden die Attribute festgelegt. Danach folgt bis Zeile 22 die Initialisierung dieser Attribute und Erstellung der Klasse „Graph_Matrix“ mit maximaleKnoten als Parameter, wobei die Anzahl der Knoten auf 0 gesetzt und diese Knoten im Array „knoten“ speichert werden. Gleichzeitig wird eine zweidimensionale Matrix erstellt, damit man die besuchten Knoten verfolgen und die kürzeste Distanz, zur Berechnung des kürzesten Weges, sowie den Weg, von den man zu einem Knoten kommt, ermitteln kann. In der Methode „KnotenEinfuegen“ (Z. 24-37) wird ein neuer Knoten zur Matrix hinzugefügt, falls dieser die Anzahl der Knoten im Graphen die maximale Anzahl an Knoten nicht überschreitet und nicht schon existiert. Die private Methode „KnotenNummer“ sucht einen Knoten anhand seiner Bezeichnung und gibt seinen Wert wieder. Der Wert -1 steht für nichts welches heißt, dass der Knoten nicht gefunden wurde. In den Zeilen 51-57 wird durch die Methode „KnotenBezeichnerGeben“ der Name eines Knoten anhand seiner Position wiedergegeben. Wenn auf dieser Position jedoch nichts existiert, wird “ “ ausgeworfen. Die Methode „KanteEinfuegen“ (Z. 59-70) ermittelt die Nummern der Knoten und fügt eine Kante ein, damit man den Weg-/Kantenwert zwischen zwei Knoten herausfindet. Dies funktioniert auch nur, wenn beide Knoten existieren und Knoten(vor) und Knoten(nach) nicht das gleiche sind. Bei der Methode „Ausgeben“ wird in den Zeilen 72 bis 90 eine Adjazenmatrix ausgegeben. Durch die Methode „KnotenAnzahlgeben“ wird nur die Anzahl der vorhandenen Knoten angegeben. Danach kann in Z.98-106 die Länge eines Weges mithilfe der Methode „KanteGewichtGeben“ ausgegeben werden, indem die Nummern der Start- und Endknoten ausgesucht werden und dann das Gewicht (Länge) der Kante ermittelt wird. Die private Methode „Besuchen“ (Z. 108-123) setzt den aktuellen Knoten als „besucht“ und gibt die Bezeichnung aus. Außerdem durchläuft man alle Nachbarknoten in der Matrix und überprüft, ob eine Kante zum Nachbarknoten existiert und ob dieser noch nicht besucht wurde. Falls die Bedingungen erfüllt sind, wird die Methode für den Nachbarknoten aus und wiederholt sich rekursiv, bis alle Nachbarknoten besucht wurden. Dies heißt dann, dass der aktuelle Knoten fertig besucht wurde. Die Methode „TiefenSuche“ (Z. 125-137) startet eine Tiefensuche im Graphen und beginnt bei dem Knoten mit angegebener Startbezeichnung und ruft währenddessen die Methode „Besuchen“ auf, um eine systematische Untersuchung aller möglicher Pfade im Graphen durchzuführen. Der

nicht besuchte Knoten mit der minimalen Distanz kann durch die Methode „MinKnoten“ wiedergegeben werden. Die Methode „KuerzesterWeg“ (Z. 155-203) wird aufgerufen, um den Kürzesten Weg zwischen einem Startknoten und einem Zielknoten zu finden. Dabei wird eine ArrayList „Weg“ initialisiert, um den kürzesten Weg zu speichern. Es werden alle Knoten auf nicht „besucht“ gesetzt, wobei die Knoten alle die größtmögliche Distanz besitzen. Außerdem wird der Abstand zum Startknoten auf „0“ gesetzt. Später werden alle umliegenden Nachbarknoten betrachtet und springt zu dem Knoten mit der kürzesten Distanz und vorher noch nicht besucht wurde. Danach wird die neue Distanz zu dem umliegenden Knoten betrachtet, welche sich aus der aktuellen Distanz und dem Matrixwert vom umliegenden Knoten zusammensetzt. Wenn die neue Distanz kleiner als die Abzweignummer (Matrixposition/Distanz vom allerersten Startpunkt), dann wird die Abzweignummer der neuen Distanz gleichgesetzt und gleichzeitig zum neuen Knotenpunkt. Dieser Vorgang wird so lange wiederholt, bis alle umliegenden Knoten betrachtet wurden. Danach wird eine fertige Liste erstellt