## Farbsensor:

Zuallererst werden der EV3, der Port, der Farbsensor und der Sampleprovider importiert. Die Attribute „Farbsensor“, „Colorprovider“, „Colorsample“ und „Color“ werden festgelegt, um in der Klasse den Port vom s3 den Farbsensor zuweisen zu können. Dabei wird der Colorprovider mit dem Farb-ID-Code vom Farbsensor gleichgesetzt, um anhand dieses Wertes im Colorsample die Länge der Liste zu bekommen. Die Methode „Readcolor“ gibt die gemessene Farbe zurück. Die Methode „Measure Color“ ist das gleiche, als ob man den Farbsensor aufruft. Nachdem eine Sache ausgeführt wurde schließt sich außerdem der Farbsensor.

## Liniensensor:

Es werden der EV3 von Lejos, der Port vom Brick, die Klasse „Mindsensorlinereader“, „Sampleprovider“, „Tasten“ und „Screen“ importiert. Dabei sind die Attribute „Line“, „Lineprovider“, „Linesample“, „Sample“, „Using“, „Off“, „Strecke”, „Außerhalb”, „Keys“, und „LCD“ eingeführt worden. Mit lineprovider holt man die Daten aus dem S4 Port, um diese dem Liniensensor zuzuweisen. Man setzt auch den lineprovider auf „benutzt“, weist die Werte zu, konfiguriert schwarz und weiß und schließt den Liniensensor da er sonst nicht funktioniert. Die Methode „Setusing“ setzt den Input in using ein. Es wird außerdem bei der Methode „readoff“ die Zahl -1 gesendet, wenn man zu weit links von der Linie ist, die Zahl 0 wenn man auf der Linie ist und 1 wenn man zu weit rechts ist. Um zu messen ob sich der Sensor links recht oder sich in der Mitte der Linie befindet, benutzt man die Methode „Calculateoff“.

## Test:

Es wird Lejos, der Zugriff auf die Buttons, die Klasse „Regulatedmotor“, „Delay“, und eine Java Array-List importiert. Die Attribute „Linie“ (Liniensensor), „Color“ (Farbsensor), „Links“ (Motor), „Rechts“ (Motor) „Liste“ (Dijkstra; GRAPH_MATRIX), „Weg“ (Adapter Weg), „Start (Startpunkt), „Ende“ (Endpunkt), „Counter“ (auf welchem Knoten gerade), „List“ (Liste von Motoren zur Synchronisation) werden deklariert. Die Methode „Fahren“ lässt den linken mit dem rechten Motor synchronisieren. Beide bewegen sich nach vorne, wodurch die Synchronisation beendet wird. Nach einer 0,25 Sekunden Pause werden rechts und links wieder synchronisiert und wieder gestoppt. Die nächste Methode ist gleich nur dass der Input in Millisekunden angegeben wird. Die Methode „LinksAbbiegen“ lässt die linke Seite des Motors nach hinten und die rechte Seite für 725 Millisekunden bewegen. Bei „RechtsAbbiegen“ ist der Vorgang gleich nur mit dem Unterschied, dass links nach vorne und rechts nach hinten bewegt wird. Um abbiegen zu können gibt die Methode „GiveDirection“ einen Buchstaben für jeden Knoten aus. Wenn dieser „L“ ist, dann fährt der Roboter nach links, wenn dieser „r“ ist, dann nach rechts und für „m“ fährt man geradeaus. Falls man zu weit rechts von der Linie steht, dann dreht sich das Gefährt bei „rechtskorrigieren“ leicht nach links und fährt 100 Millisekunden geradeaus. Im Gegensatz dazu wird bei

„linkskorrigieren“ nach rechts gedreht und 100 Millisekunden gefahren, falls sich das Fahrzeug zu weit links befindet. Die „Main“ Methode lässt die Testklasse aufrufen. Dies bewirkt, dass solange der Escape Button nicht gedrückt wird und der Counter nicht negativ ist, ein „On the way“ erscheint. Außerdem wird überprüft, ob Start- und Endknoten das gleiche sind. Sind diese gleich, ist man am Ziel angekommen und der Vorgang wird abgebrochen. Sind diese jedoch nicht gleich, schaut man auf welcher Farbe man steht. Für grün wird „GiveDirection“ ausgeführt und für schwarz wird betrachtet, wo man sich auf der Linie befindet. Wenn sich der Roboter in der Mitte der Linie befindet, dann fährt man nach vorne, wobei wenn man sich zu weit rechts befindet, „linkskorrigieren“ ausgeführt wird, oder zu weit links „rechtskorrigieren“ ausgeführt wird.

## AdapterWeg:

Man importiert eine Array-List und deklariert die Attribute „MaxKnoten“ (Max. Anzahl der Knoten), „Dijkstra“ (Zugriff auf Dijkstra Code/ „GRAPH_MATRIX“), „Richtung“ (Zugriff auf „Richtung“ Code), „Start“ (Startknoten), „End“ (Endknoten) und „Map“ (Karte). In dieser Klasse wird ein Import für Start- und Endknoten verlangt. Außerdem werden „Map“, „Start“ und „End“ initialisiert. Die Methode „ListeGeben“ gibt eine Array-List zum Weg von Start zu Ende. Durch „StartGeben“ wird der Startknoten und durch „EndGeben“ der Endknoten bestimmt.

## Motor:

Es wird „RegulatedMotor“, der EV3, der Port und der „LargeRegulatedMotor“ importiert. Das Attribut „Motor“ wird deklariert. Diese Klasse hat einen Input wobei „Input“ sagt welchen Port man dem Motor zuweist. Bei „forward“ bewegt sich der Motor nach vorne. Bei „backware“ nach hinten. Und abschließend stoppt der Motor bei aufführung von „Stop“.