**1.3.4. Befehlsreferenz**

Für die Befehle in Java Karol wurde der Roboter „karol“ genannt.

|  |  |  |
| --- | --- | --- |
| ***Aktionen*** | | |
| Robot Karol | Java Karol | Bedeutung |
| Schritt | karol.Schritt(); | Karol macht einen Schritt in Blickrichtung. |
| Linksdrehen | karol.LinksDrehen(); | Karol dreht um 90° nach links. |
| Rechtsdrehen | karol.RechtsDrehen(); | Karol dreht um 90° nach links. |
| Hinlegen | karol.Hinlegen(); | Karol legt einen Ziegel vor sich hin. |
| Aufheben | karol.Aufheben(); | Karol hebt den Ziegel auf, der vor ihm liegt. |
| MarkeSetzen | karol.MarkeSetzen(); | Karol setzt an seiner Position eine Marke. |
| MarkeLöschen | karol.Markelöschen(); | Karol löscht eine Marke an seiner Position. |
|  | karol.QuaderAufstellen(); | Karol stellt vor sich einen Quader auf. |
|  | karol.QuaderEntfernen(); | Karol entfernt einen Quader, der vor ihm steht. |
| Ton | karol.Tonerzeugen() | Karol gibt einen Ton aus. |
|  | karol.MeldungAusgeben("…"); | Karol gibt eine Meldung aus. |

|  |  |  |
| --- | --- | --- |
| ***Logische Abfragefunktionen*** | | |
| Robot Karol | Java Karol | Karol meldet true |
| IstWand | karol.IstWand(); | wenn er vor einer Wand oder einem Quader steht. |
| NichtIstWand | !karol.IstWand(); | wenn er nicht vor einer Wand oder einem Quader steht. |
| IstZiegel | karol.IstZiegel(); | wenn er vor einem Ziegel steht. |
| NichtIstZiegel | !karol.IstZiegel(); | wenn er nicht vor einem Ziegel steht. |
|  | karol.IstZiegelLinks(); | wenn links von ihm ein Ziegel ist. |
| IstNorden | karol.IstBlickNorden(); | wenn er in Richtung Norden schaut. |

|  |  |  |
| --- | --- | --- |
| ***Kontrollstrukturen*** | | |
| wenn *Bedingung* dann  *Anweisung*  …  \*wenn | if (*Bedingung*)  {  *Anweisung*;  …  } | einseitige Verzweigung |
| wenn *Bedingung* dann  *Anweisung*  …  sonst  *Anweisung*  …  \*wenn | if (*Bedingung*)  {  *Anweisung*;  …  }  else  {  *Anweisung*;  …  } | zweiseitige Verzweigung |
| wiederhole *Anzahl* mal  *Anweisung*  …  \*wiederhole | for (int i=0; i<*Anzahl*; i++)  {  *Anweisung*  …  } | Zählschleife |
| wiederhole solange Bedingung  *Anweisung*  …  \*wiederhole | while (*Bedingung*)  {  *Anweisung*;  …  } | Wiederholung mit Anfangsbedingung |
| wiederhole  *Anweisung*  …  \*wiederhole solange Bedingung | do  {  *Anweisung*;  …  }  while (*Bedingung*) | Schleifen mit Endbedingung |