

## Übung: 6.3 Roboter

Schreiben Sie eine Klasse **Roboter**, die die Position eines Roboters in einem Koordinatensystem speichern soll.

```
public class Roboter
{
    // Attribute
    .....
    .....

    // Konstruktoren
    public Roboter( double xStart, double yStart )
    {
        .....
        .....
    }

    public Roboter()
    {
        .....
        .....
    }

    // Getter
    public double getX()
    {
        .....
    }

    // Setter
    public void setX(double xNeu)
    {
        .....
    }

    // allgemeine Methode
    public void geheNachRechts()
    {
        .....
    }

    // überladene Methode
    public void geheNachRechts(int strecke)
    {
        .....
    }
}
```

```

    public boolean stehtRechtsVon( Roboter vergleichsroboter )
    {
        .....
    }
}

```

## *TDD – test-driven development*

Wir wollen nun die Klasse **Roboter** schrittweise erstellen und sowohl manuell wie auch automatisiert testen.

### 1. Schritt

Erstellen Sie nun die Klasse mit den Attributen, den beiden Konstruktoren und den Gettern.

**Manueller Test:** Übersetzen Sie nun die Klasse und erzeugen Sie Objekte mit verschiedenen Konstruktoren.

Überprüfen Sie nun, ob die Attribute richtig belegt wurden, in dem Sie die Attribute im Objekt-Inspektor betrachten. Außerdem testen Sie die beiden Getter-Methoden manuell und überprüfen Sie die korrekte Rückgabe der Methoden.

**Automatisierter Test:** Übersetzen Sie nun die Klasse **TesteKonstruktorUndGetter** und führen Sie anschließend den Test aus.

### 2. Schritt:

Ergänzen sie nun in der Klasse **Roboter** die beiden Setter-Methoden. Testen Sie die Korrektheit ihrer neuen Methoden wie im oberen Schritt beschrieben, in dem sie zuerst manuell testen. Anschließend übersetzen Sie die Testklasse **TestSetter** und führen den automatisierten Test durch.

### 3. Schritt



Ergänzen Sie die Methoden **geheNachRechts()**, **geheNachLinks()**, **geheNachOben()**, **geheNachUnten()** und testen Sie zuerst manuell und anschließend mit der Testklasse **TesteGehenOhneParameter**.

### 4. Schritt

Ergänzen Sie die Methoden **geheNachRechts(int)**, **geheNachLinks(int)**, **geheNachOben(int)**, **geheNachUnten(int)** und testen Sie zuerst manuell und anschließend mit der Testklasse **TesteGehenMitParameter**.

### 5. Schritt

Ergänzen Sie die Methoden **stehtRechtsVon(Roboter)**, **stehtLinksVon(Roboter)**, **stehtUeber(Roboter)**, **stehtUnter(Roboter)** und testen Sie zuerst manuell und anschließend mit der Testklasse **TesteStehtRelativZu**.

	Name:	AuP	Objektorientierte Programmierung <b>KLASSENKONZEPT IN JAVA</b>	
---	-------	-----	---	---

## Ergänzungsübungen für Schüler mit Vorkenntnissen

### 6. Schritt

Ergänzen Sie in der Klasse einen sogenannten Copy-Konstruktor **Roboter(Roboter r)**, der von einem Roboter **r**, der als Parameter übergeben wird, seine Koordinaten übernimmt und ein neues Roboter-Objekt mit den identischen Koordinaten erzeugt. Testen Sie zuerst manuell und anschließend mit der Testklasse **TesteCopyKonstruktor**.

### 7. Schritt

Ergänzen Sie in der Klasse die Methoden **abstandVomUrsprung()**, **distanz(Roboter)** und **triff(Roboter)**. Diese können Sie mit der Testklasse **TesteDistanzermittlung** überprüfen.

Tipp: Mit **Math.sqrt( x )** können Sie in Java die Wurzel von **x** berechnen.

### 8. Schritt

Ergänzen Sie nun die Methode **equals(Object o)**, die in vielen Java-Bibliotheken benötigt wird, und deren fehlerfreie Implementierung äußerst schwierig ist. Leider wird diese Methode in realen Projekten immer wieder falsch realisiert. Für werdende Java-Profis empfehle ich dazu im Buch „Der Weg zum Java-Prof“ die Seiten 165 bis 171.

Testen Sie zuerst manuell und anschließend mit der Testklasse **TesteEquals**.