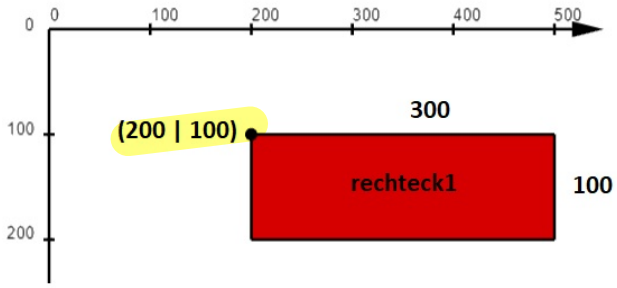
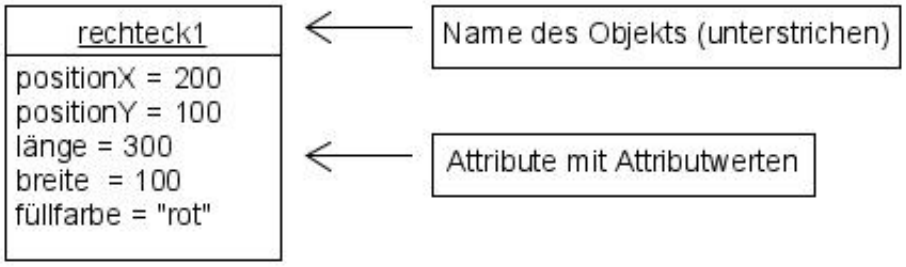
2 Theoretische Grundlagen der Objektorientierung

2.1 Objekte

Wie der Name schon sagt, sind so genannte Objekte bei der Objektorientierung ein zentraler Begriff. Objekte können allgemein Dinge (z.B. Räume, Autos,...), Lebewesen (z.B. Menschen, Tiere,...) oder Begriffe (z.B. Konten, Buchungen,...). Jedes Objekt kann durch Eigenschaften beschrieben werden, diese Eigenschaften eines Objekts werden **Attribute** genannt. Zum Beispiel sind dies bei einem Rechteck länge, breite, füllfarbe,... (Hinweis: Attribute werden in der Programmierung klein geschrieben)
Ein konkreter Wert für ein Attribut wird **Attributwert** genannt. Betrachtet man folgendes beispielhafte Objekt `rechteck1`:



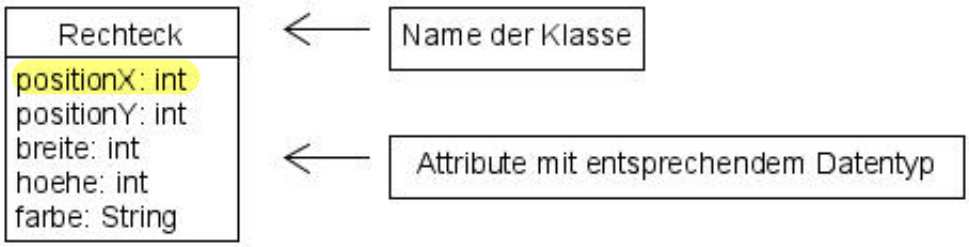
So könnte dieses Objekt durch folgende Attribute beschrieben werden:
`positionX`, `positionY` (für die linke obere Ecke), `länge`, `breite` und `füllfarbe`.
Die Attributwerte für das Objekt `rechteck1` wären dann für positionX 100, für positionY 200, für länge 300, für breite 100 und für füllfarbe „rot“.
Um diese Informationen übersichtlich darzustellen, verwendet man in UML **Objektkarten bzw. Objektdiagramme**:



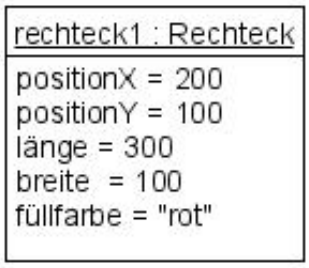
Hinweis: Sollten Sie Objektkarten per Hand zeichnen, achten Sie darauf, dass Sie den Objektnamen unterstreichen und nicht durchstreichen.

2.2 Klassen

Klassen sind Bau- bzw. Konstruktionspläne für gleichartige Objekte. Sie dienen dabei als Vorlage zur Erzeugung von neuen Objekten. Eine Klasse besitzt unter anderem eine Liste von Attributen mit deren dazugehörigem Datentyp. Jedes Objekt derselben Klasse besitzt die gleichen Attribute, die Attributwerte der Objekte können sich aber unterscheiden. Die Informationen einer Klasse werden übersichtlich in einer **Klassenkarte** dargestellt. Beispielsweise wäre ein Bauplan für das Objekt `rechteck1` die Klasse `Rechteck` mit folgender Klassenkarte:



Dabei werden die Objektkarten um den dazugehörigen Klassennamen wie folgt erweitert:



Wenn Sie bei den nachfolgenden Aufgaben Objektkarten bzw. Klassenkarten zeichnen müssen, verwenden Sie das Werkzeug UMLet.

Aufgaben:

1. (a) Erklären Sie kurz in eigenen Worten den Unterschied zwischen Attribut und Attributwert.

   Attribut: Eigenschaft (Haare)

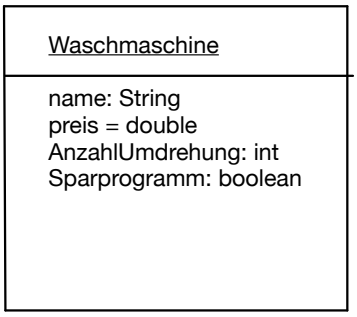
   Attributwert: Konkreter Wert: (Haare= blond)

   (b) Erklären Sie kurz in eigenen Worten den Zusammenhang zwischen einem Objekt und einer Klasse.

   Die Klasse benötigt ein Objekt, um eben das Obj in „Wirklichkeit“ abzubilden

2. Gegeben ist folgender Ausschnitt eines Angebots für eine Waschmaschine.

Klassendiagramm



Objektdiagramm

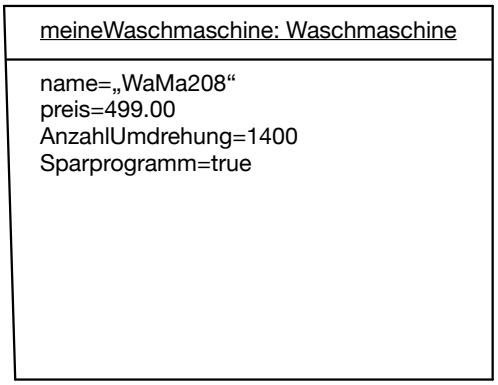


Abbildung 1: Schulbuch Informatik 1A (Klett Verlag), S.14

(a) Zeichnen Sie eine Klassenkarte für die angebotene Waschmaschine mit vier Attributen. Dabei sollen alle Attribute einen unterschiedlichen Datentyp besitzen.

(b) Zeichnen Sie mit der erstellten Klassenkarte eine entsprechende Objektkarte für die angebotene Waschmaschine.

2.3 Methoden

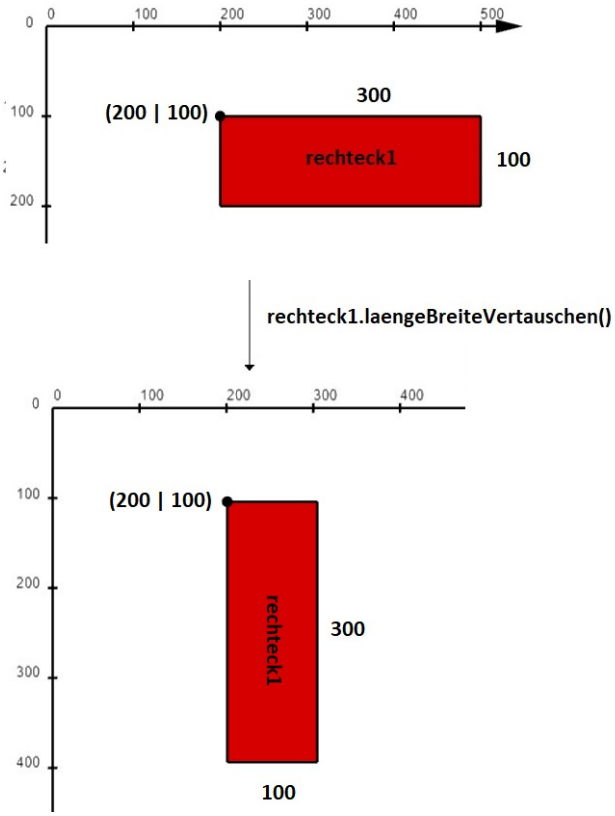
Methoden sind „Tätigkeiten“, die Objekte ausführen können. Diese Tätigkeiten werden in der Klasse des Objekts definiert. Im Folgenden soll die Klasse `Rechteck` um verschiedene Methoden erweitert werden.

2.3.1 Länge und Breite vertauschen

Möchte man bei Rechtecken die Länge und Breite vertauschen, so muss in der Klasse `Rechteck` dafür eine Methode `laengeBreiteVertauschen` (Methodennamen sollten in der Regel wie Attributsnamen mit einem Kleinbuchstaben beginnen) definiert werden. Soll dann anschließend das Objekt `rechteck1` diese Methode ausführen, so erfolgt die Anweisung über den **Punktoperator**:

```
rechteck1.laengeBreiteVertauschen()
```

rechteck1.verschieben(100, 100)



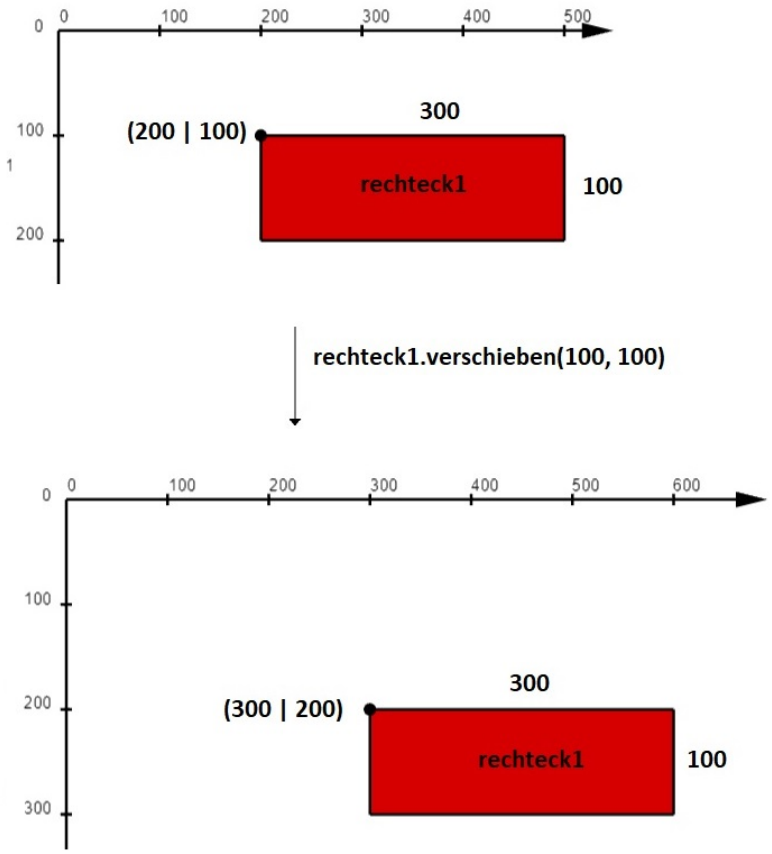
Parameter

Objekt kann so viele Parameter haben, wie es will.

2.3.2 Verschieben

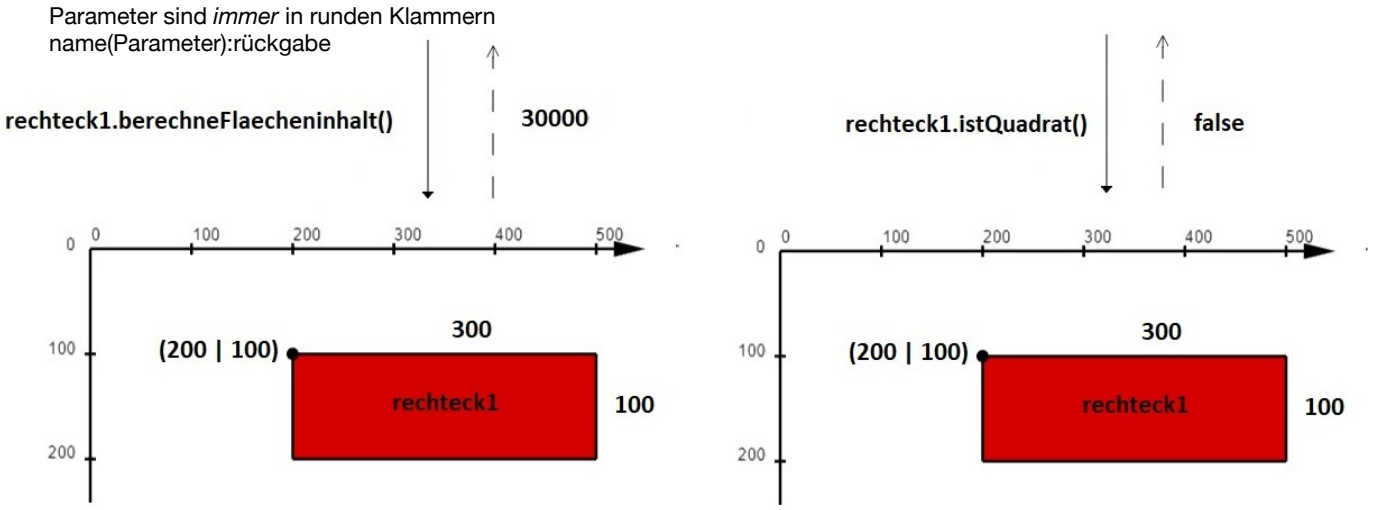
Sollen nun zusätzlich Rechteck-Objekte verschoben werden können, so muss die Klasse `Rechteck` eine Methode `verschieben` bereit stellen. Der Unterschied zum ersten Beispiel ist, dass hierfür noch weitere Informationen übergeben werden müssen: Um wie viel das Rechteck in x- bzw. y-Richtung verschoben werden soll. Diese übergebenen Werte werden auch **Übergabeparameter** genannt und stehen beim Methodenaufruf in den runden Klammern. Mehrere Parameter werden durch Kommas getrennt bzw. bleiben die Klammern leer, falls es keine Übergabeparameter gibt (siehe Beispiel 1). Ein Aufruf könnte dann beispielsweise wie folgt aussehen:

```
rechteck1.verschieben(100, 100)
```



2.3.3 Informationen über Objekte

Häufig benötigt man in der Programmierung Informationen über Objekte. Bei Rechtecken könnte dies beispielsweise die Größe des Flächeninhalts sein oder ob es sich bei dem Rechteck um ein Quadrat handelt. Um an diese Informationen zu kommen, werden häufig Methoden verwendet.
Dies bedeutet, dass diese Methode einen Wert für eine Weiterverarbeitung zurückgeben müssen. Das heißt, eine Methode für die Größe des Flächeninhalts würde ein Wert vom Datentyp `int` (**Rückgabedatentyp**) zurückgeben, eine Methode, ob ein Rechteck ein Quadrat ist, ein Wert vom Datentyp `boolean`, welcher für die Weiterverarbeitung gespeichert werden kann:



Codebeispiel:

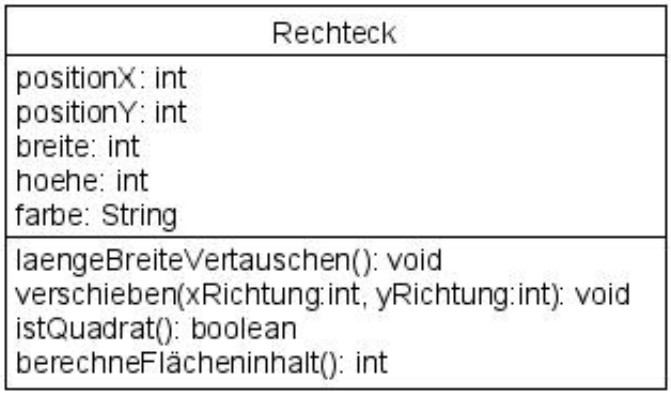
```
int flaecheninhalt = rechteck1.berechneFlaecheninhalt()
boolean istQuadrat = rechteck1.istQuadrat()
```

Bei Methoden muss immer ein Rückgabedatentyp angegeben werden. Gibt eine Methode keinen Wert zurück, so verwendet man als Rückgabedatentyp das Schlüsselwort **`void`**. Diesen Rückgabedatentyp `void` würde man bei den beiden vorherigen Methoden `laengeBreiteVertauschen` und `verschieben` verwenden.

Die Information über die Methoden einer Klasse werden nach folgendem Muster in die Klassenkarte mit aufgenommen:

Methodenname(Übergabeparameter mit Datentyp): Rückgabedatentyp

Den Teil Methodenname(Übergabeparameter mit Datentyp) nennt man dabei **Methodensignatur.**
Die bisherige Klassenkarte der Klasse `Rechteck` wird damit um die neu definierten Methoden wie folgt erweitert:



Aufgaben:

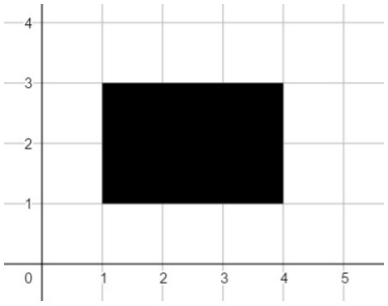
1. Nennen Sie einen Vorteil, wenn für die Größe des Flächeninhalts kein Attribut angelegt wird, sondern eine Methode mit Rückgabedatentyp verwendet wird. Unter welchen Umständen wäre ein Attribut für die Größe des Flächeninhalts besser geeignet als eine Methode?

   Der Flächeninhalt ergibt sich aus 2 Eigenschaften des Rechtecks => Berechnet sich daraus; A=LxB
   Damit man nicht immer die Methode aufrufen muss
   Arbeitsspeicher & Prozessor werden hier belastet

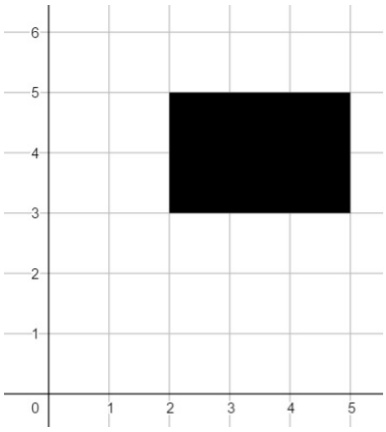
2. Im Folgenden sehen Sie ein Objekt *r1* zu vier verschiedenen Zeitpunkten (Zeitpunkt 1 bis Zeitpunkt 4). Dieses Objekt besitzt dabei folgende Methodensignaturen:

   - setFüllfarbe(farbe:String)
   - setBreite(breite:String)
   - setLinienart(linienart:String)
   - drehen(drehwinkel:int)
   - verschieben(xRichtung:int, yRichtung:int)

   Geben Sie jeweils einen Methodenaufruf in der Punktnotation an, der zwischen den abgebildeten Zeitpunkten geschehen ist:

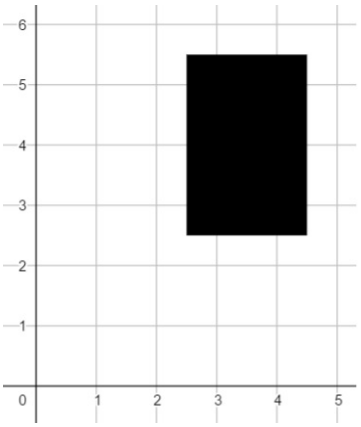


Zeitpunkt 1

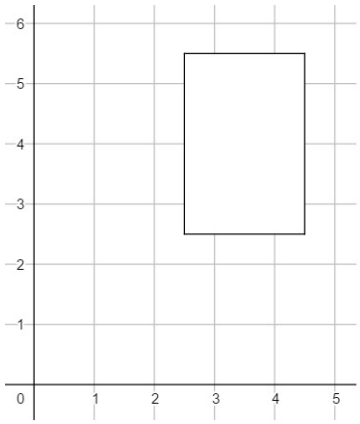


verschieben(1,2)

Zeitpunkt 2



Zeitpunkt 3



Zeitpunkt 4

Methodenaufruf von Zeitpunkt 1 nach Zeitpunkt 2:

```
r1.verschieben(1,2);
```

Methodenaufruf von Zeitpunkt 2 nach Zeitpunkt 3:

```
r1.drehen(270);
```

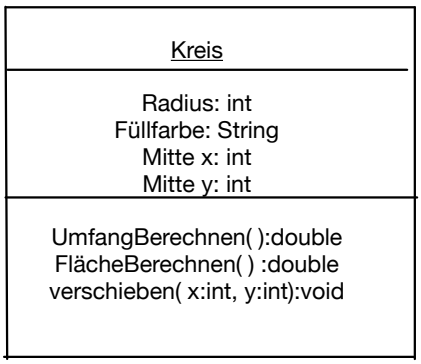
Methodenaufruf von Zeitpunkt 3 nach Zeitpunkt 4:

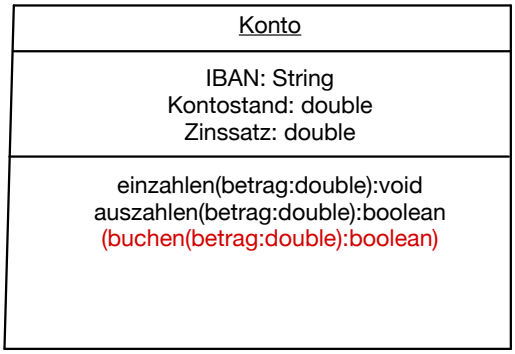
```
r1.setFüllFarbe(“weiss“);
```

3. Erstellen Sie für folgende Situationen jeweils eine UML-Klassenkarte mit Attributen und Methoden.

   (a) Ein Kreis besitzt einen ganzzahligen Radius und einen Mittelpunkt mit ebenfalls ganzzahligen Koordinaten. Außerdem besitzt er eine Füllfarbe. Kreise sollen verschoben werden können, sowie Information über den Umfang und Flächeninhalt bereitstellen.

   (b) Ein Konto besitzt einen Kontostand, einen Zinssatz (in Prozent) sowie eine IBAN. Bei einem Konto sollen Beträge eingezahlt werden können, sowie Beträge abgehoben werden können. Das Abheben soll aber nur passieren, wenn der Kontostand größer gleich als der zu abhebende Betrag ist. Ob der Abhebevorgang dementsprechend erfolgreich war, soll durch eine entsprechende Rückgabe signalisiert werden.





3 Umsetzung in Java (ohne Methoden)

3.1 Von der Klassenkarte zur Java-Klasse

Die Klassenkarte eines Rechtecks soll nun in Java umgesetzt werden. Die Methoden werden dabei zu einem späteren Zeitpunkt umgesetzt.

Die Attribute werden mit ihrem Datentyp dabei zeilenweise wie folgt für eine Jave-Klasse übersetzt:

```
public class Rechteck{
  int positionX;
  int positionY;
  int breite;
  int hoehe;
  String farbe;
}
```

Attribute mit gleichem Datentyp können dabei in einer Zeile zusammengefasst werden:

```
public class Rechteck{
  int positionX, postionY, breite, hoehe;
  String farbe;
}
```

Aus Gründen der Übersichtlichkeit ist davon aber eher abzuraten.

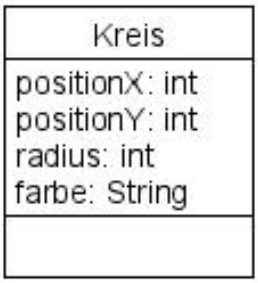
Hinweis: Achten Sie bei der Umsetzung in den folgenden Aufgaben darauf, dass sie Attribute exakt so schreiben, wie sie in der Klassenkarte stehen.

Aufgaben:

1. Erstellen Sie ein neues Java-Projekt `Einstieg_OOP` und dort im `src`-Ordner ein neues Package `figuren`. Achten Sie darauf, dass Sie mindestens Java-Version 17 für das Projekt eingestellt haben.

2. Erstellen Sie in diesem Package eine neue Klasse `Rechteck` mit den obigen Attributen.

3. Erstellen Sie in diesem Package ebenfalls eine weitere Java-Klasse für folgende Klassenkarte (ohne Methoden).



3.2 Erzeugen von Objekten: Der Konstruktor und Punktoperator

3.2.1 Erste Schritte

Um nun Figuren zeichnen zu lassen, müssen Objekte der jeweiligen Figur-Klasse erzeugt werden.
Ein neues Objekt der Klasse Rechteck `rechteck1` kann mit dem Schlüsselwort **`new`** wie folgt erzeugt werden:

```
Rechteck rechteck1 = new Rechteck();
```

Mit `new Rechteck()` wird dabei der so genannte **Konstruktor** der Klasse Rechteck aufgerufen, welcher ein neues Objekt der Klasse `Rechteck` im Arbeitsspeicher „erzeugt“. Der Konstruktor ist dabei eine Art Vorschrift, wie ein Objekt dieser Klasse erzeugt werden soll.

Auf die Attributwerte des Objekts `rechteck1` kann mit dem **Punktoperator** folgendermaßen zugegriffen werden.

```
rechteck1.breite
rechteck1.farbe
...
```

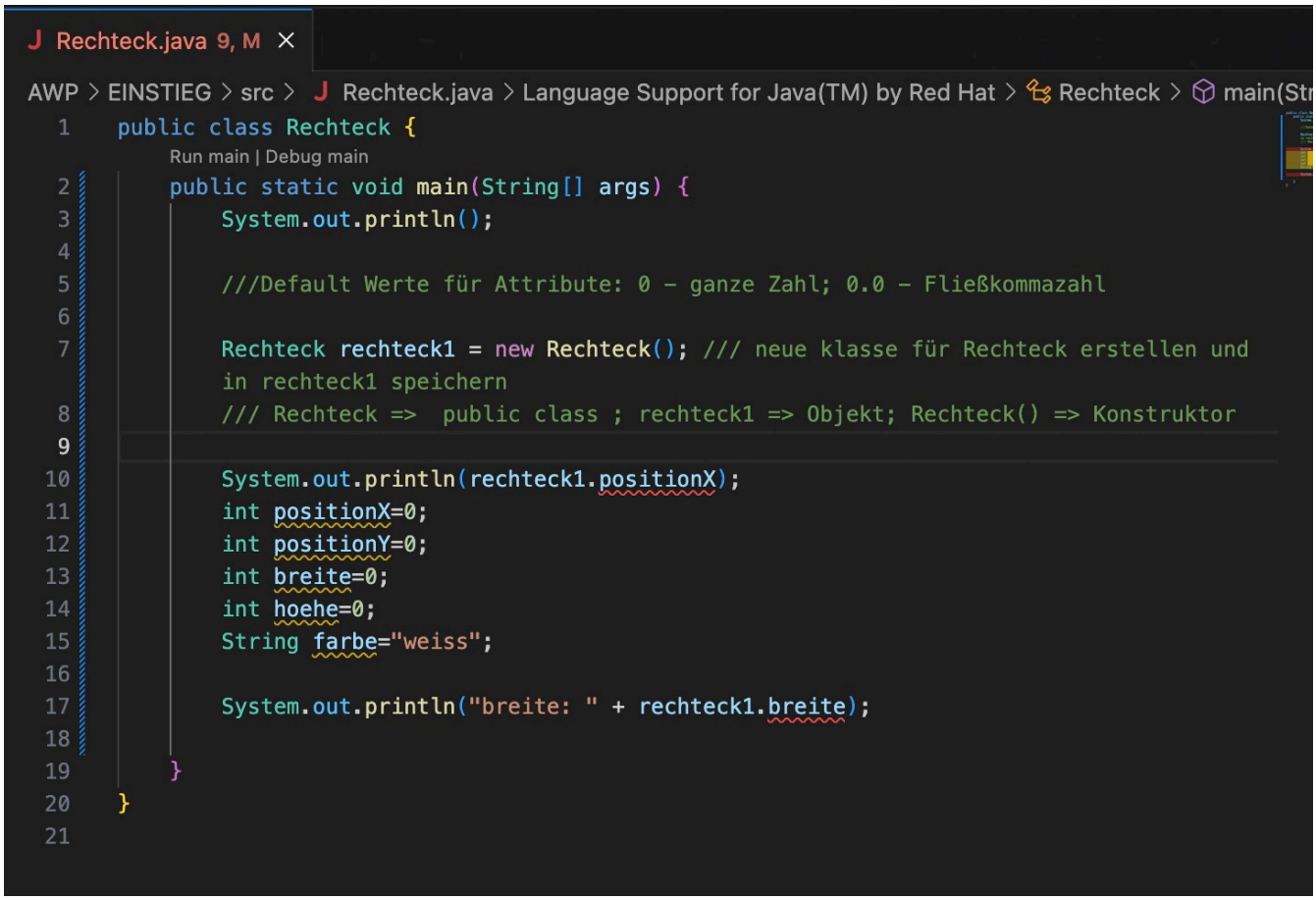
Allgemein: `Objektname.Attributname`

Aufgaben:

1. Erstellen Sie in Ihrem Package eine neue Klasse `Test` mit einer `main-Methode`.
2. Erzeugen Sie in dieser `main-Methode` ein neues Objekt `rechteck1` der Klasse `Rechteck`.

3. Geben Sie nun alle Attributwerte des Objekts `rechteck1` auf der Konsole aus. Was fällt Ihnen auf?

Konstruktion heißt wie die Klasse



3.2.2 Genauere Betrachtung des Konstruktors

Wird für eine Klasse kein expliziter Konstruktor angegeben, so werden für die Attribute bei der Objekterzeugung Standardwerte verwendet (0 für `int`, `false` für `boolean`, `null` für `String`,...). Um nun für jedes Attribut sinnvolle Attributwerte bei der Objekterzeugung festzulegen, muss in der Klasse `Rechteck` explizit ein Konstruktor angegeben werden. Dieser könnte beispielsweise folgendermaßen aussehen:

```
Rechteck(){
   positionX = 200;
   positionY = 150;
   farbe = "rot";
   //... Weitere Zuweisungen
}
```

Wird nun ein Objekt der Klasse `Rechteck` erstellt, so besitzt das Objekt von Anfang an den Attributwert 200 für das Attribut `positionX`, den Wert „rot“ für das Attribut `farbe`,..., wichtig ist dabei:

Konstruktoren müssen exakt so geschrieben werden wie der Klassenname.

Konstruktoren werden als Methode ohne Rückgabetyp in Klassenkarten gekennzeichnet:

Aufgaben:

1. Was ist die Aufgabe eines explizit angegebenen Konstruktors?

   Wird benutzt, um das Objekt einer Klasse zu initialisieren

2. Ergänzen Sie die Klasse `Rechteck` um einen Konstruktor, so dass bei der Objekterzeugung die linke obere Ecke bei (200|150) liegt und das Objekt die Höhe 100, die Breite 300 und die Farbe rot besitzt.

3. Damit Ihre Implementierung grafisch getestet werden kann, müssen sie die jar-Datei `shapes.jar` aus dem Klassenlaufwerk in Ihr Projekt einbinden. Gehen Sie dabei in Eclipse wie folgt vor:
   (a) Erstellen Sie mit einem Rechtsklick auf Ihren Projektordner einen neuen Ordner (`New` → `Folder`) mit dem Namen `lib`.
   (b) Kopieren Sie die Datei `shapes.jar` aus dem Klassenlaufwerk in diesen Ordner. (Hinweis: Eclipse unterstützt dies per Drag and Drop).
   (c) Nun muss diese Datei dem `Classpath` hinzugefügt werden. Dies funktioniert folgendermaßen:
   Rechtsklick in Eclipse auf die Datei `shapes.jar` → `Build Path` → `Add to Build Path` auswählen.

   Hinweis: Mögliche Anleitung für IntelliJ und VS-Code.

4. Erstellen Sie in der `main`-Methode Ihrer Klasse `Test` aus dem vorherigen Aufgabenblock, analog zum Objekt `rechteck1` ein neues Objekt `leinwand` der Klasse `Leinwand`.
   Hinweis: Damit Sie Objekte der Klasse `Leinwand` in einer Klasse verwenden können, müssen Sie folgende Import-Anweisung vor der Klassendefinition Ihrer Klassen einfügen:

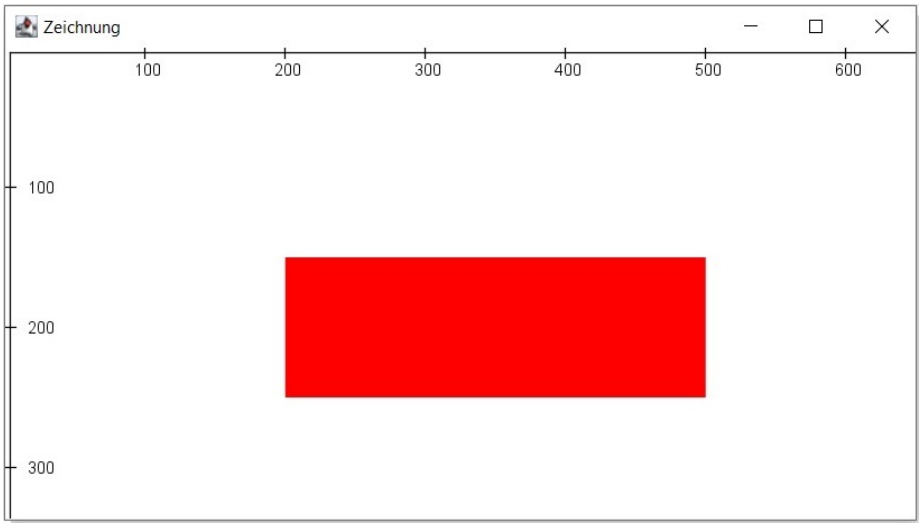
```
import ack.shapes.Leinwand;

public class Test{
...
}
```

5. Testen Sie Ihren Konstruktor der Klasse `Rechteck`, indem Sie Ihr Objekt `rechteck1` mit der Methode `zeichne` des Objekts `leinwand` zeichnen lassen:

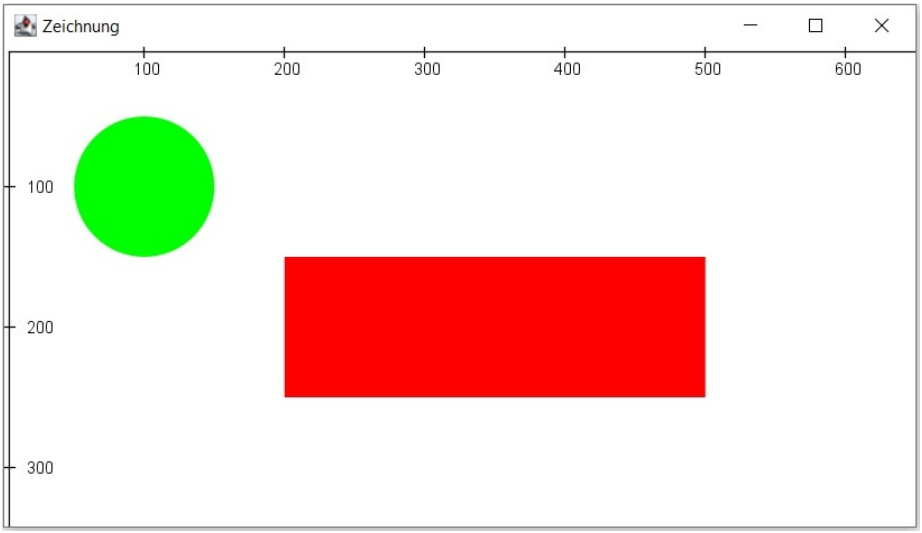
   ```
   leinwand.zeichne(rechteck1);
   ```

   Wenn Sie nun das Programm ausführen, sollte dabei das Rechteck folgendermaßen gezeichnet werden:



6. Erstellen Sie nun analog einen Konstruktor für die Klasse `Kreis`, so dass bei der Objekterzeugung der Mittelpunkt bei (100|100) liegt und das Objekt den Radius 50 und die Farbe „gruen“ besitzt.

7. Testen Sie Ihren Konstruktor, indem Sie in der `main`-Methode Ihrer Klasse `Test` ein weiteres Objekt `kreis1` der Klasse `Kreis` erzeugen und es ebenfalls mit der Methode `zeichne` des Objekts `leinwand` zeichnen lassen.
   Dabei sollte nach der Ausführung die Zeichnung wie folgt aussehen:

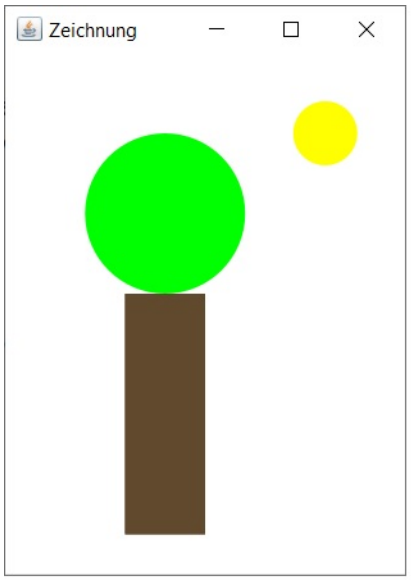


3.2.3 Erste Zeichnung

Nun können Sie eine erste richtige Zeichnung anfertigen.

Aufgaben:

1. Erstellen Sie eine neue Klasse `ErsteZeichnung` mit einer `main`-Methode.
2. Legen Sie in der `main`-Methode geeignete Objekte der Klassen `Leinwand, Rechteck` und `Kreis` an, so dass folgendes Bild gezeichnet wird. Überlegen Sie sich mit einer Skizze passende Attributwerte für Ihre Objekte und verwenden Sie den Punktoperator, um die passenden Attributwerte zuzuweisen.

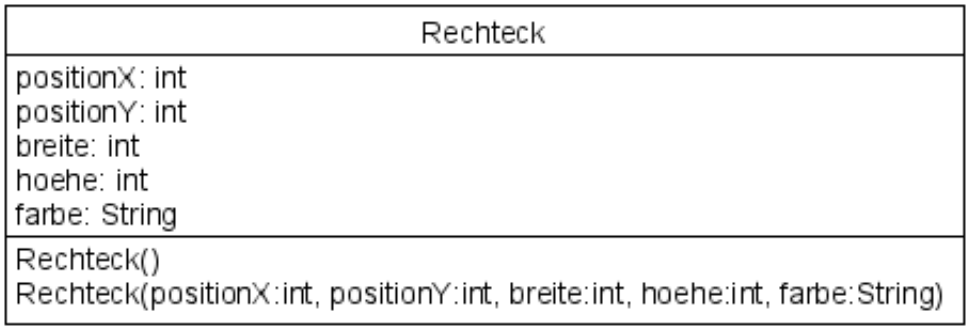


3.2.4 Mehrere Konstruktoren für eine Klasse

Beim Zeichnen ist das passende Setzen jedes einzelnen Attributwertes für jedes neue Objekt ziemlich umständlich. Einfacher ist es, bei der Objekterzeugung die passenden Werte direkt dem Konstruktor als Parameter zu übergeben. Anstelle den bisherigen Konstruktor zu ändern, kann man dafür einen weiteren Konstruktor definieren:

```
Rechteck(int positionX, int positionY, int breite, int hoehe,
          String farbe){
   this.positionX = positionX;
   this.positionY = positionY;
   //Weitere Zuweisungen...
}
```

Hierbei ist zu beachten, dass das Schlüsselwort `this` notwendig ist. Mit diesem Schlüsselwort stellt man eine Referenz zur aktuellen Instanz her, d.h. mit `this.positionX` stellt man einen Bezug zum Attribut `positionX` der Klasse her. Dies muss in diesem Fall gemacht werden, um den Namenskonflikt aufzuheben bzw. da der übergebene Parameter das Klassenattribut verdeckt.
Besitzt eine Klasse mehrere Konstruktoren, nennt man dies **überladen von Konstruktoren**. Wichtig ist dabei, dass sich alle Konstruktoren einer Klasse in den Übergabeparametern unterscheiden.
Dieser Konstruktor wird ebenfalls in die Klassenkarte mit aufgenommen:



Aufgaben:

1. Die Klasse `Leinwand` besitzt neben dem Konstruktor ohne Parameter einen weiteren Konstruktor. Beschreiben Sie kurz den weiteren Konstruktor.

2. Ergänzen Sie die Klasse `Rechteck` um einen weiteren Konstruktor `Rechteck(int positionX, int positionY, int breite, int hoehe, String farbe)`, welcher die Attributwerte auf die entsprechenden übergebenen Werte setzt. Ergänzen Sie die Klasse `Kreis` um einen analogen Konstruktor.

3. Benutzen Sie nun die neuen Konstruktoren, um den Programmcode aus dem vorherigen Aufgabenblock in der Klasse `ErsteZeichnung` zu vereinfachen.

4. Vereinfachen Sie in den Klassen `Rechteck` und `Kreis` die Konstruktoren ohne Parameter, indem diese Konstruktoren den jeweilige Konstruktor mit Parametern aufrufen. Dies ist ebenfalls mit dem Schlüsselwort `this` möglich.