II.2. Objekte, Klassen und Methoden

- 1. Grundzüge der Objektorientierung
- 2. Methoden, Unterprogramme und Parameter
- 3. Datenabstraktion
- **4. Konstruktoren**
- 5. Vordefinierte Klassen

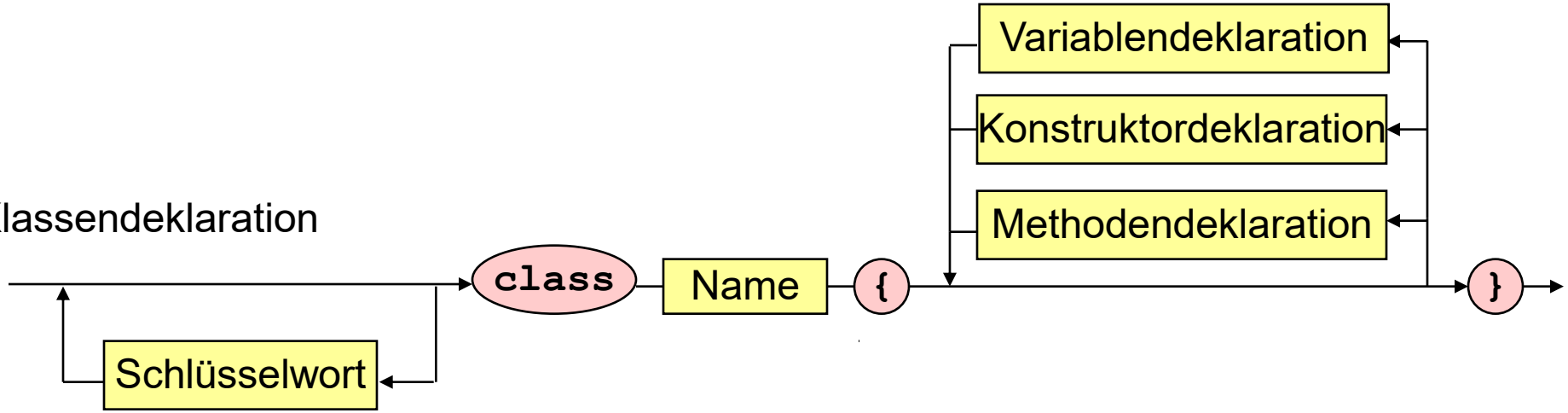
Konstruktoren

```
public class Rechteck {

   //Objektattribute
  double laenge, breite;       int strichstaerke;

  //Konstruktor 1
  public Rechteck () {
     laenge = breite = 1.0;    strichstaerke = 1;
  }

 //Konstruktor 2
 public Rechteck (double x, double b) {
     laenge = x;  breite = b;  strichstaerke = 1;
 }
...
}
```

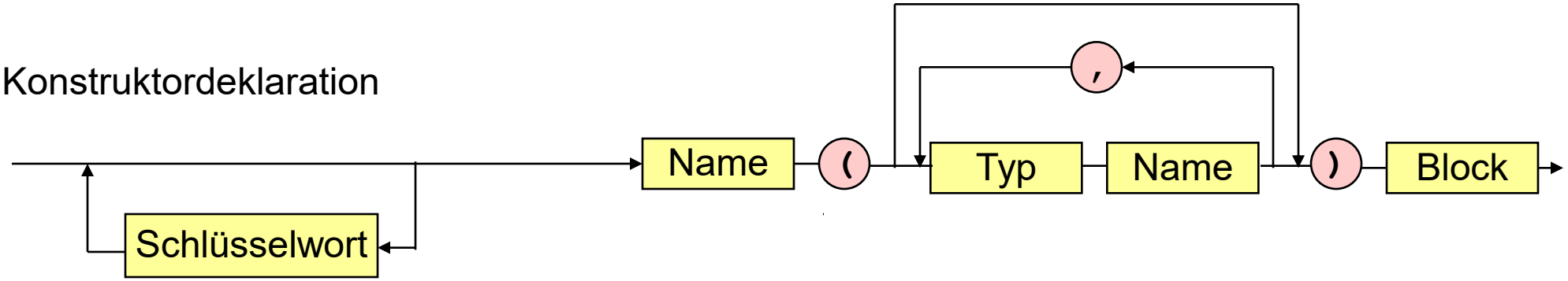
```
Rechteck r = new Rechteck ();
Rechteck s = new Rechteck (2.1,1.5);
```

Konstruktordeklaration

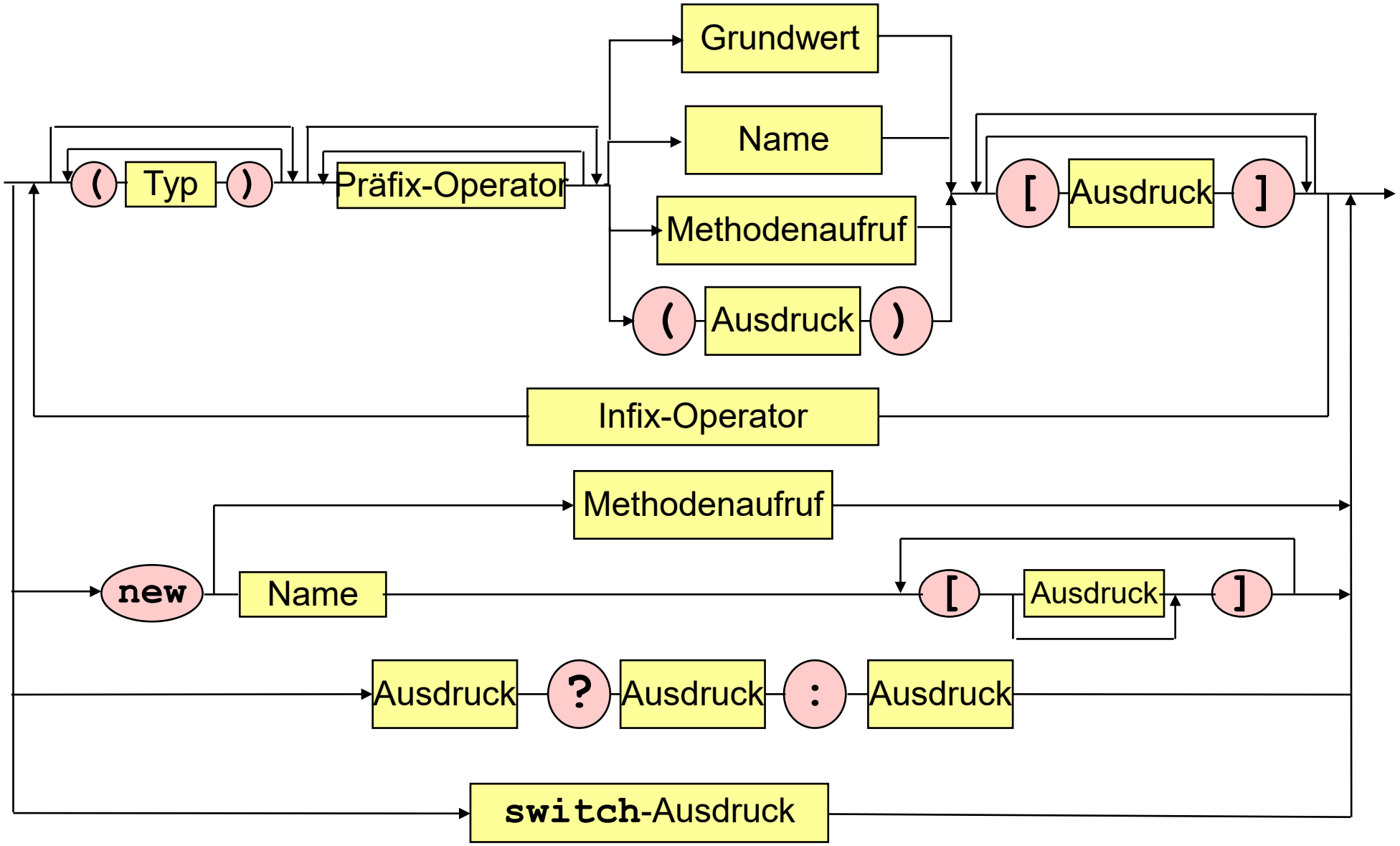
Klassendeklaration



Konstruktordeklaration



Ausdruck


Grundwert
Name
Methodenaufruf
(Ausdruck)
(Typ)
Präfix-Operator
[Ausdruck]
Infix-Operator
Methodenaufruf
new
Name
[Ausdruck]
Ausdruck ? Ausdruck : Ausdruck
switch-Ausdruck

Überladene Konstruktoren

```
1 public Rechteck () {
      laenge = breite = 1.0;                strichstaerke = 1; }

2 public Rechteck (double x, double b) {
      laenge = x;  breite = b;              strichstaerke = 1; }

3 public Rechteck (double kantenlaenge) {
      laenge = breite = kantenlaenge;       strichstaerke = 1; }

4 public Rechteck (int s) {
      laenge = breite = 1.0;                strichstaerke = s; }

5 public Rechteck (int... a) {
      laenge = a[0]; breite = a[1]; strichstaerke = 5;     }
```

```
Rechteck r = new Rechteck ();      Konstr. 1
Rechteck s = new Rechteck (1,2);   Konstr. 2
Rechteck t = new Rechteck (3.0);   Konstr. 3
Rechteck u = new Rechteck (3);     Konstr. 4
Rechteck v = new Rechteck (1,2,3); Konstr. 5
```

vararg Methoden möglichst nicht überladen!

Vararg-Methode wird nur benutzt, wenn andere Methoden nicht passen.

Konstruktoren mit Selbstverweis

```
public Rechteck () {
     laenge = breite = 1.0;              strichstaerke = 1;
}

public Rechteck (double laenge, double breite) {
     this.laenge = laenge;
     this.breite = breite;
     strichstaerke = 1;
}

public Rechteck (double laenge) {
     this.laenge = breite = laenge;
     strichstaerke = 1;
}

public Rechteck (int strichstaerke) {
     laenge = breite = 1.0;
     this.strichstaerke = strichstaerke;
}
```

Kopier - Konstruktor

```
public Rechteck (Rechteck original) {

     if    (original != null) {
           laenge = original.laenge;
           breite = original.breite;
           strichstaerke = original.strichstaerke;
     }
     else  { // kein Original vorhanden
           laenge = breite = 1.0;
           strichstaerke = 1;
     }

}
```

Records

```
public record Rechteck (double laenge, double breite,
                        int strichstaerke) {


}
```

```
public class Rechteck {
  double laenge, breite;      int strichstaerke;

  public Rechteck (double laenge, double breite,
                   int strichstaerke) {
     this.laenge = laenge;
     this.breite = breite;
     this.strichstaerke = strichstaerke;
  }

  public String toString () {...}
}
```

Records

```
public record Rechteck (double laenge, double breite,
                        int strichstaerke) {

     public Rechteck {
          IO.println("Neues Rechteck");
     }

}
```

```
Rechteck r = new Rechteck(1.0, 2.0, 3);
```

Neues Rechteck

```
IO.println(r.laenge());
```

Rechteck[laenge=1.0, breite=2.0, strichstaerke=3]

```
IO.println(r);
```