```
//Überladener Konstruktor
public Kreis(double radius, int xPos, int yPos, Color farbe)
{
    this.Radius = radius;
    this.XPos = xPos;
    this.YPos = yPos;
    this.Farbe = farbe;
}
```

Der Operator this erlaubt den Zugriff auf die Felder der eigenen Klasse. Dadurch kann sicher zwischen den Feldern der Klasse und den Übergabeparametern der Funktion unterschieden werden, wenn diese den gleichen Namen besitzen.

Der überladene Konstruktor kann z.B. durch

```
Kreis K2 = new Kreis(42.1, 200, 60, Color.Red);
```

aufgerufen werden und erlaubt das gleichzeitige Anlegen eines Objektes und das Zuweisen von Werten an die Felder des Objektes.

3.5. Wert- und Verweistypen

Im Allgemeinen unterscheidet man bei höheren Programmiersprachen zwischen so genannten „primitiven“ Datentypen (z.B.: int, short, float, …) und zusammengesetzten Datentypen (z.B.: struct und class).

In C# ist es außerdem wichtig, Daten nach der Art der Speicherung zu unterscheiden.

Werttypen speichern ihre Werte direkt in den Variablen (siehe Abbildung 3.2 oben). Der Compiler reserviert dem Datentyp entsprechend Speicherplatz auf dem Stack.

In C# sind die primitiven Datentypen, aber auch die Struktur Werttypen.

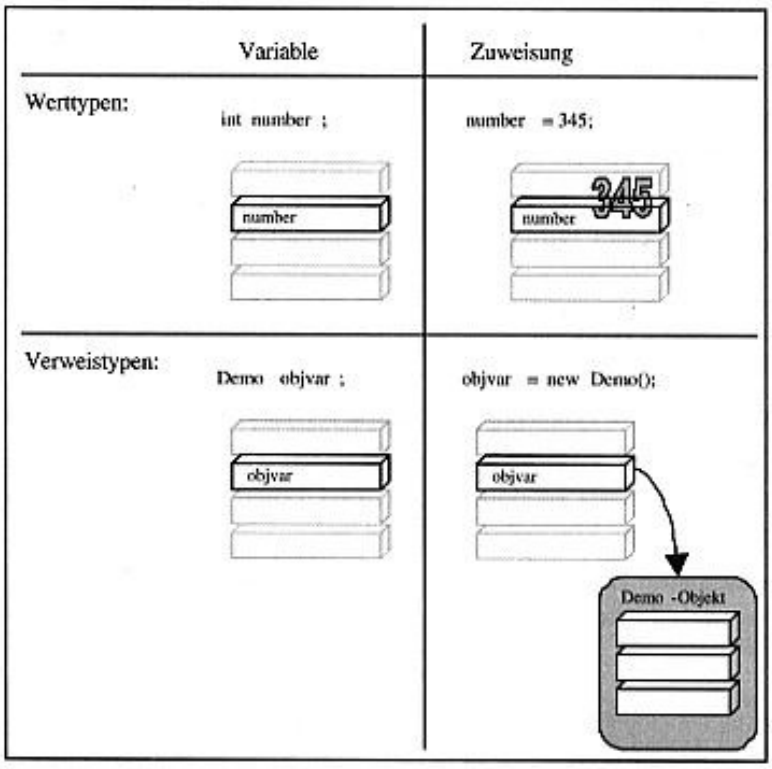

Abbildung 3.2 Vergleich Wert- und Verweistyp[1]

[1] LOUIS Dirk, STRASSER Shinja, LÖFFELMANN Klaus: Visual C# 2005, Das Entwicklerbuch

Verweistypen speichern ihre Werte, die wegen ihrer Komplexität allgemein als Objekte bezeichnet werden, im frei allokierbaren Speicherbereich (Heap). Zusätzlich wird am Stack nur eine Variable angelegt, die auf das Objekt im Heap verweist.

In Abbildung 3.2 unten wird auf der linken Seite mit der Anweisung

```
Demo objvar;
```

eine Variable objvar am Stack angelegt.

Auf der rechten Seite wird mittels

```
new Demo();
```

ein Objekt im Heap angelegt und der Variablen objVar durch

```
objVar = new Demo();
```

ein Verweis auf dieses Objekt zugewiesen. objVar zeigt jetzt auf das Objekt im Heap.

In Abbildung 3.3 ist folgendes Beispiel dargestellt:

Durch die Programmzeilen

```
AClass objJim = new AClass();
AClass objTim = new AClass();
```

werden wie in Abbildung 3.3 links dargestellt im Stack zwei Verweisvariablen objJim und objTim angelegt, die auf zwei verschiedene Objekte der Klasse AClass im Heap verweisen.

Was aber passiert durch folgende Zuweisung?

```
objTim = objJim;
```

Die Variablen verweisen jetzt auf das gleiche Objekt im Heap, das zweite Objekt bleibt verwaist im Speicher zurück (siehe Abbildung 3.3 rechts).

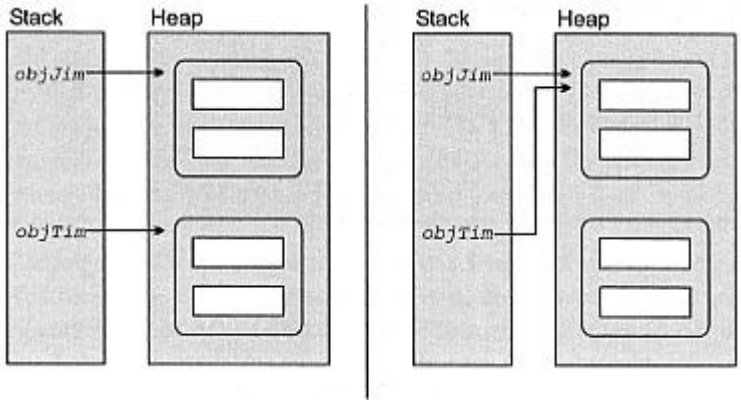

Abbildung 3.3 Zuweisung für Verweistypen

3.6. Ausnahmebehandlung (Exception handling)

Mit Hilfe der Ausnahmebehandlung sollen Laufzeitfehler z.B. durch fehlerhafte Benutzereingaben verhindert werden.

Die allgemeine Struktur der Ausnahmebehandlung sieht folgendermaßen aus: