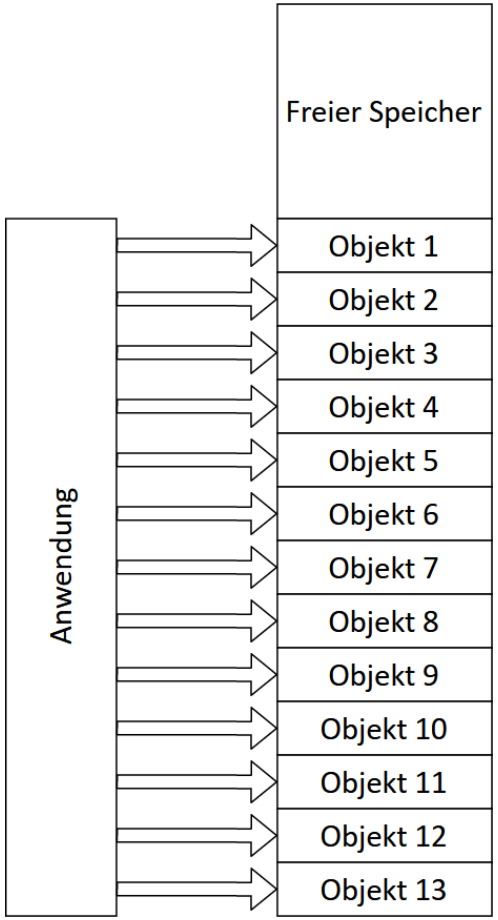
Garbage Collection, Statische Member, Werte und Referenzen, Null

Garbage Collection

- verwaltet die Belegung und Freigabe von Arbeitsspeicher
- überprüft ob Objekte noch verwendet werden
- beginnt diese zu „zerstören“ um den Speicher freizugeben
- läuft komplett eigenständig im Hintergrund

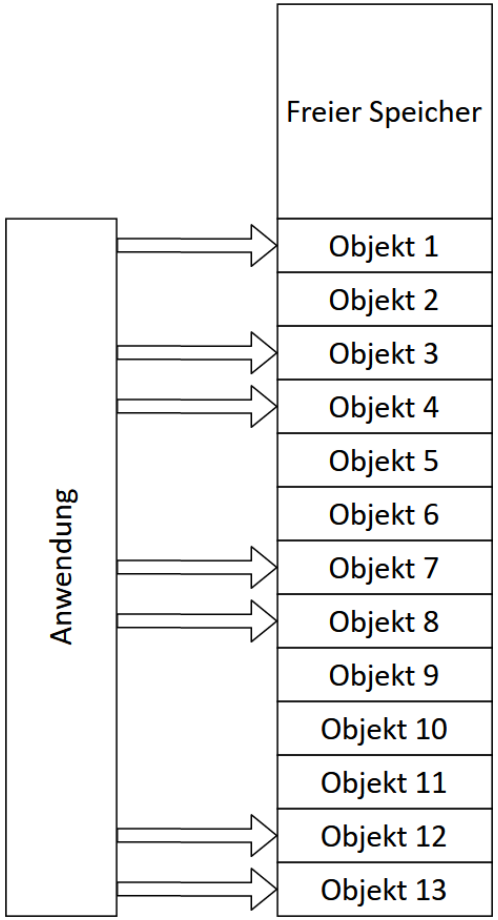
- kümmert sich nur um verwaltete Ressourcen (!)

Garbage Collection



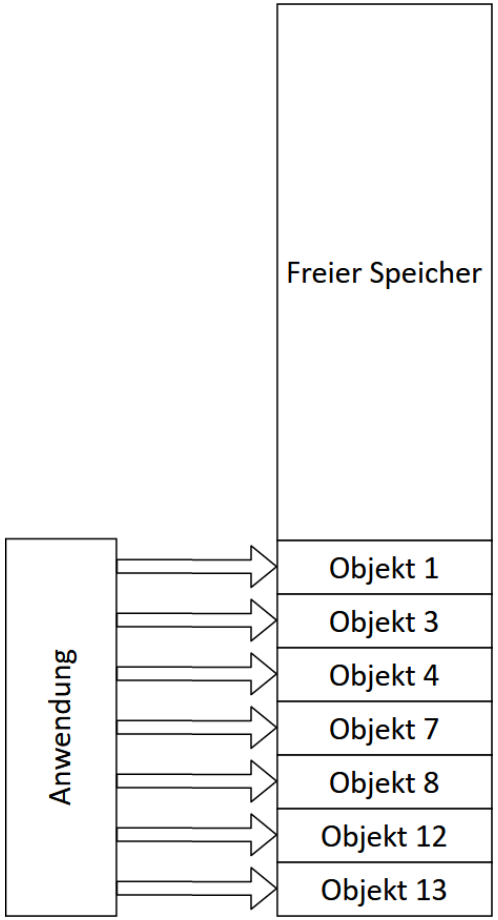
- die Anwendung hat mehrere Objekte im Speicher, welche alle noch benötigt werden

Garbage Collection



- die Anwendung benötigt einige Objekte nicht mehr, diese sind aber noch im Speicher vorhanden

Garbage Collection



- die GC filtert genau diese Objekte heraus und „zerstört“ sie, sodass der Speicher wieder freigegeben wird

Destruktor (Finalizer)

- wird aufgerufen wenn ein Objekt vom Garbage Collector zerstört wird
- kann nur einmal pro Klasse festgelegt werden
- Funktionsname besteht aus einer Tilde (~) und dem Klassennamen
- kann selber Anweisungen ausführen

```
~Person()
{
    Console.WriteLine("Destruktoraufruf: das Objekt wird nun zerstört");
}
```

Statische Member

- gelten für die Klasse und nicht für ein Objekt dieser Klasse
- Zugriff erfolgt über den Klassenname
- statische Member können nicht auf nicht-statische Member der Klasse zugreifen
- Jede Eigenschaft (Property) existiert nur ein Mal für die Klasse („systemweit“)
- klassische Beispiele

```
Console.WriteLine("Text");

string eingabe = Console.ReadLine();

DateTime heute = DateTime.Now;
```

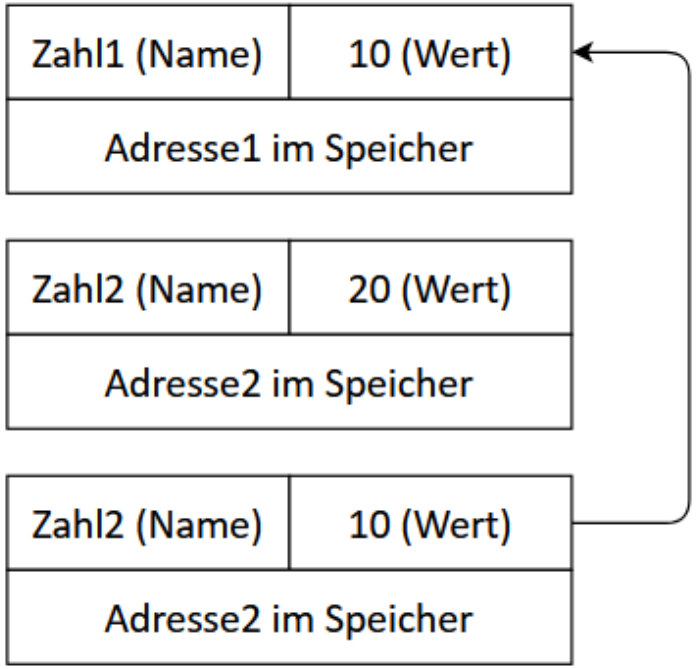
Wertetypen und Referenztypen

- Wertetypen
  - werden mit einer festen Adresse im Speicher hinterlegt
  - der Wert wird beim Ändern an genau der Stelle im Speicher geändert
- Referenztypen
  - werden einmalig im Speicher hinterlegt
  - bei Änderung bekommen sie eine neue Adresse
  - können einer bereits genutzten Adresse zugewiesen werden

Wertetypen und Referenztypen

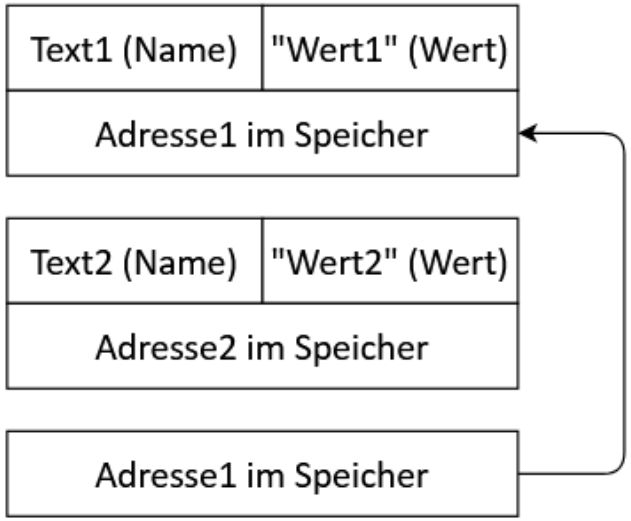
Wertetypen

```
int Zahl1 = 10;
int Zahl2 = 20;
Zahl2 = Zahl1;
```



Referenztypen

```
string Text1 = "Wert1";
string Text2 = "Wert2";
Text2 = Text1;
```



Schlüsselwort - ref

- gleicher Hintergrund wie out
- Parameter muss zuvor initialisiert werden
- Wert des Parameters kann direkt verwendet werden
- Aufruf ebenfalls mit Schlüsselwort

Schlüsselwort - ref

```
int addiere(int zahl1, int zahl2, ref int anzahlAdditionen)
{
    int summe = zahl1 + zahl2;
    anzahlAdditionen += 1;

    return summe;
}

int anzahl = 0;
int ergebnis = 0;
ergebnis = addiere(ergebnis, 3, ref anzahl); //ergebnis = 3 anzahl = 1
ergebnis = addiere(ergebnis, 8, ref anzahl); //ergebnis = 11 anzahl = 2
ergebnis = addiere(ergebnis, 45, ref anzahl); //ergebnis = 56 anzahl = 3
```

Null

- Standardwert wenn eine Variable keinen Wert hat
- Ausnahmen: int, double, bool, …
- Mit einer if überprüfen