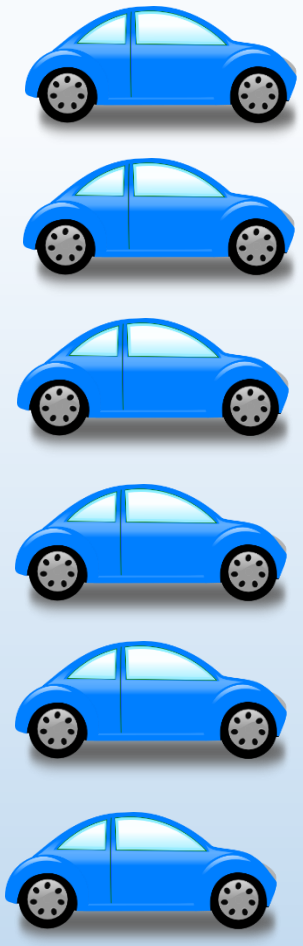
Klassen und Objekte II

tAuto



at[0]

at[1]

at[2]

at[3]

at[4]

at[5]

at: array of tAuto

- Hinzufügen neuer Autos
- Löschen von Autos
- auf HDD speichern
- von HDD laden
- Suchen

etc….

Kombinieren von Daten (Array) und Methoden

```
tGarage =class
private
 at : array of Tauto;
public
 procedure dazu( a: tAuto);
 procedure weg ( i: integer);
 function  suche(s: string): integer;
 procedure speichern(pfad: String);
 procedure laden (pfad :string);
 …
end;
```

Klassen und Objekte II

```
tGarage =class
private
 at : array of Tauto;
public
 procedure dazu( a: tAuto);
 procedure weg ( i: integer);
 function  suche(s: string): integer;
 procedure speichern(pfad: String);
 procedure laden (pfad :string);
 …
end;
```

hat – Beziehung (**AGGREGATION**)

1 ——— **1..n**

```
tAuto=class
private
 marke, typ:string;
 bj, km:integer;
…
```

tGarage ist für die Verwaltung der Instanzen von tAuto mittels eines Arrays verantwortlich

-> besitzt Methoden zum Modifizieren des Arrays (dazu, weg)
-> besitzt Methoden zum Anzeigen der Elemente des Arrays (suche, ausgabe…)
-> besitzt Methoden zum Speichern und Laden des Arrays

-> wenn tGarage erzeugt wird, muss das Array auf 0 gesetzt werden
(wichtige Voraussetzung um später neue Autos hinzuzufügen)

-> wir erledigen dies, indem wir den Konstruktor von tGarage ÜBERSCHREIBEN

Wie erinnern uns: Jede Klasse hat automatisch den Konstruktor **create** . Diesen können wir anpassen:

Deklaration:

```
tGarage =class
private
 at : array of Tauto;
public
 procedure dazu( a: tAuto);
 procedure weg ( i: integer);
 function  suche(s: string): integer;
 procedure speichern(pfad: String);
 procedure laden (pfad :string);
 constructor create;
 …
end;
```

Implementierung:

```
constructor tGarage.create;
begin
  inherited create;
  setlength(at, 0);
end;
```

vom „Original“ – create übernommen:
Speicher für Objekt wird angelegt

zusätzliche Aktion:
Array wird auf 0 gesetzt

Vererbung

```
tAuto=class
private
   marke, typ:string;
   bj, km:integer;
   preis:real;
public …
end;
```

Mutterklasse
- enthält die Basiseigenschaften

```
tLKW=class(tAuto)
private
  achsen:integer;
  gesgewicht: real;
public
 …
end;
```

```
trennwagen=class(tAuto)
private
  ps, vmax:integer;
  gewicht: real;
  rennklasse:string;
public
 …
end;
```

Kindklassen
- **enthält alle Felder und Methoden der Mutterklasse**
- **kann zusätzlich eigene Felder und Methoden haben**
- **kann Methoden der Mutterklasse ändern**

DIe Mutterklasse aller Klassen heißt **TObject.** Sie besitzt lediglich die Methode **create**. (-> reserviert Speicher)

Klassen und Objekte II

Aufgabenstellung 1:

1) nutze die Anwendung „Garage“ als Ausgangsbasis
2) leite von der Mutterklasse tAuto die Klasse tRennwagen ab
3) Leite tRennstall als Tochterklasse von tGarage ab (soll ein Array von Rennwagen verwalten)
4) Schreibe die Methoden zu tRennstall (dazu, weg..) passend um.
5) Ergänze um das Feld *Ranking:real* und die Methoden *aufsteigen* und *absteigen*

Aufgabenstellung 2:

Wir brauchen für diverse Spiele immer wieder die Möglichkeit, Highscores einzutragen. Bauen wir also eine Klasse *tHighscore*, die wir mit allen möglichen anderen Anwendungen benutzen können.

1) Lege zunächst die Form der Einträge fest, die im Highscore gemacht werden sollen.
   Lege hierfür die Klasse *TEintrag* an.
2) Verwalte eine Liste von 10 Einträgen in der Klasse *tHighscore.*
3) Lege die Methoden *hinzufuegen(e:Teintrag) , anzeigen:string* und *sortieren* an.