Aufbau einer einfachen Datenstruktur

Diese Beschreibung soll vor allem helfen, die Unterschiede zwischen einer Datenstruktur in C und in den objektorientierten Sprachen zu verstehen.

➔ Im Gegensatz zu Datenstrukturen, die immer nur Daten enthalten, können Klassen **Daten UND Funktionen** als Elemente enthalten.

In C: -> die Verwendung von „struct“
In C++ & Java -> Definition einer Klasse mit Attributen und Methoden.

1 Der Aufbau in C

In C wird die Datenstruktur wie folgt deklariert (ein Beispiel):

```
struct  spielkarte
{
      char farbe[20];
      int wert;
};
```

Jetzt werden wie gewohnt, Variablen diesem Typ zugewiesen:

```
struct spielkarte  karte1, karte2;
```

Statt dass die Variable „karte1“ vom Typ integer oder character ist, ist es nun vom komplexen Datentyp „spielkarte“.

Der Zugriff erfolgt über die einzelnen Felder der Struktur:

```
printf(„die farbe ist %s\n“, karte1.farbe);
```

➔ Spielkarte wird wie ein neuer Typ verwendet (also, wie ein string oder char oder int, …..)

2 Der Aufbau in C++:

Von der Struktur zur **Klasse** ist es nur ein kleiner Schritt:

```
class  spielkarte
{
     public:
          char farbe[20];
          int wert;
};
```

Das ist bereits eine Klasse, die wir wie eine Datenstruktur anlegen. Das nennt man dann „instanzieren“.

Der Zugriff auf die Attribute in der Klasse geschieht wiederum mit dem „.“-Operator. Also genau gleich wie beim Zugriff auf eine Struktur in C.

```
spielkarte s;

s.wert = 10;
```

3 public oder private ?

Bei der Verwendung einer struct oder einer Klasse mit öffentlichen („public“) Daten kann jeder Benutzer die Daten ändern.
Wenn man das nicht möchte, dann sollte man die Attribute als „private“ deklarieren.
Man definiert dann Funktionen (Methoden) in der Klasse, welche den Zugriff erlauben:

```
class  spielkarte
{
     private:
          char farbe[20];
          int wert;


     public:
          char *get_farbe()  {return farbe;}
          int get_wert() {return wert;}

};
```

Speziell ist hier, dass die Funktion „get_farbe“ einen Pointer auf den String hat. (C++ verwendet aber eine Standard String-Klasse, was somit die Handhabung von Strings wesentlich vereinfacht.)

4 Der Aufbau in Java:

Der Aufbau ist sehr ähnlich wie bei C++:

```
public class Spielkarte{
     private String farbe;
     private Integer wert;
     ….

     public String getFarbe(){
          return farbe;
     }

     public Integer getWert(){
          return wert;
}
```

5 Klasse und Objekte

Am besten merken wir uns den Unterschied so: eine Klasse ist die Vorlage (wie ein Template) für das entsprechende Objekt. Gearbeitet wird aber mit Objekten.

Wenn ich also eine Klasse verwenden möchte, dann muss ich das Objekt „instanziieren“. In C ist das eine einfache Zuweisung. Aber bei objektorientierten Sprachen verwendet man einen Konstruktor, um das Objekt zu „aktivieren“.

```
Spielkarte myCard = new Spielkarte();
```

Die Variable „myCard“ ist vom Typ (= Klasse) „Spielkarte“. Damit diese Variable als Objekt existiert, wird mittels „new“ das Objekt erzeugt.
Das geschieht mit dem **Konstruktor** (also: `Spielkarte()`).

Jetzt kann mein Objekt `myCard` die Attribute und die Methoden der Klasse `Spielkarte` verwenden.

Zum Vergleich in C:

```
struct spielkarte  karte1;
```