

Grundlagen der Informatik II – 11. Übung

Schwerpunkt: Wiederholung OOP, Überladung von Funktionen

Wiederholung OOP

- Grundidee: Kapselung von Daten und verwaltenden Funktionen in einem Datentypen
- Klasse: Datentyp-Konstrukt zur Kapselung (Vorlage)
 - z.B.: „Student“, „Automobil“, „Gebäude“
- Objekt: Variable vom Typ einer Klasse (konkrete Verwendung)
 - z.B.: „Student mit Matrikel 55555“, „Auto mit Kennzeichen C-23454“, „TU Chemnitz Gebäudeteil Strasse der Nationen“

- Klassendefinition:

```
class KlassenName {
    private:
        // private Attribute und Methoden
    public:
        // Konstruktor: Aufruf bei Objektdefinition
        KlassenName();
        // Destruktor: Aufruf bei Objektzerstörung/Programmende
        ~KlassenName();
        // öffentliche Attribute und Methoden
};
```

- Methodenimplementation:

```
returntyp Klassenname :: methodenname (parameterliste){
    // ...
}
```

=> Konstruktor und Destruktor ohne Rückgabotyp (returntyp) !

- Objektdefinition:

```
Klassenname objektname;
```

- Zugriff auf Methoden eines Objektes:

```
objektname . methodenname(parameter);
```

Überladung von Funktionen

- Szenario: 2 Funktionen mit gleichem Namen, aber unterschiedlichen Parametern

```
void leere(int feld[20]) { ... } // Zahlen auf 0 setzen
void leere(char feld[20]) { ... } // Zeichen auf ' ' setzen
```

- Annahme: Aufruf der Funktion leere auf folgende Weise:

```
char str[] = "Zeichenkette";
leere(str);
```

- Frage: Ist dies möglich oder wird sich der Compiler beschweren?
- Antwort: Es ist möglich. Die Funktion `void leere(char feld[20])` würde aufgerufen werden.
- Überladung von Funktionen: Erstellung mehrerer Funktionen mit dem selben Namen, allerdings unterschiedlicher Parametertypen
- Vorteil: konsistente Bezeichnungen für unterschiedliche Datentypen