Übersichtsblatt:
Generische Klasse (Templates)

Grundidee:

Wenn man eine Klasse entwickelt die später noch im Projekt verwendet werden soll, so ist es durchaus möglich, das man noch nicht genau weiß, welchen Typ die Exemplarvariablen dieser Klasse später einmal haben müssen oder es besteht sogar die Möglichkeit, dass Objekte erzeugt werden müssen, die mit mehreren Datentypen auskommen können.

Ein Ansatz ist nun, für jeden möglichen Datentyp eine eigene Klasse zu schreiben. Allerdings sind derartige Copy-and-Past-Programme unter Informatikern nicht sehr beliebt.

Einen verbesserten Ansatz bieten hier generische Klassen (auch Templates genannt). Statt den Exemplarvariablen dieser Klassen statisch einen Datentyp zuzuordnen legt man eine Variable für den Datentyp an. Welchen Typ die Exemplarvariablen letztendlich erhalten wird dann beim erzeugen des Objektes festgelegt.

Erzeugung:

Klassendeklaration:

Die Deklarationszeile einer generischen Klasse sieht wie folg aus:

```
template<typename T>class KlassenNahme:public manKannAuchErben{
```

Hierbei gibt das Stichwort `template` an dass es sich um eine generische Klasse handelt. `typename` T ist die Variable für den Datentyp der Klasse.

Deklaration der Exemplarvariablen:

Eine generische Klasse, dann wie jede andere Klasse auch ganz normal Exemplarvariablen mit festem Datentyp haben. Daneben gibt es allerdings auch noch Exemplarvariablen mit variablem Datentyp.

```
private:
        int normaleVariable;
        T variableMitVariablemDatentyp;
```

Besonderheit bei der Methoden-Deklaration:

Hier gibt es natürlich auch ganz normale Exemplarmethoden, daneben existieren aber auch noch Methoden, deren Rückgabewert einen Variablen Datentyp hat.

```
public:
        int getNormaleVariable();
        T getVariableMitVariablenDatentyp();
```

Besonderheit bei der Methoden-Implementierung:

Bei der Implementierung der Methoden ist darauf zu achten, dass vor jede Methode der Text `template<typename T> Ruckgabetyp KlassenName<T>::` zu schreiben ist.

```
template<typename T>
int KlassenName<T>::getNormaleVariable(){return normaleVariable;}

template<typename T>
T KlassenName<T>::getVariableMitVariablemDatentyp(){return variable…}
```

Zulässig Typen:

Am Ende der Implementierung werden dann noch die Typen festgelegt, die T annehmen kann.

```
template class KlassenName<int>
template class KlassenName<double>
template class KlassenName<…>
```